μSim: A goal-driven framework for elucidating the neural control of movement through musculoskeletal modeling

Muhammad Noman Almani[1,2*], John Lazzari[1,3], Andrea Chacon[4], Shreya Saxena[1,3*]

[1]Center for Neurocomputation and Machine Intelligence, Wu Tsai Institute, Yale University, New Haven, CT, USA.
[2]Department of Electrical Engineering, Yale University, New Haven, CT, USA.
[3]Department of Biomedical Engineering, Yale University, New Haven, CT, USA.
[4]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA.

*Corresponding author(s). E-mail(s): muhammadnoman.almani@yale.edu; shreya.saxena@yale.edu;

Abstract

How does the motor cortex (MC) produce purposeful and generalizable movements from the complex musculoskeletal system in a dynamic environment? To elucidate the underlying neural dynamics, we use a goal-driven approach to model MC by considering its goal as a controller driving the musculoskeletal system through desired states to achieve movement. Specifically, we formulate the MC as a recurrent neural network (RNN) controller producing muscle commands while receiving sensory feedback from biologically accurate musculoskeletal models. Given this real-time simulated feedback implemented in advanced physics simulation engines, we use deep reinforcement learning to train the RNN to achieve desired movements under specified neural and musculoskeletal constraints. Activity of the trained model can accurately decode experimentally recorded neural population dynamics and single-unit MC activity, while generalizing well to testing conditions significantly different from training. Simultaneous goal- and data- driven modeling in which we use the recorded neural activity as observed states of the MC further enhances direct and generalizable single-unit decoding. Finally, we show that this framework elucidates computational principles of how neural dynamics enable flexible control of movement and make this framework easy-to-use for future experiments.

1 Introduction

Behavior arises from the interaction among functionally and anatomically distinct entities of the brain, body and physical environment. The computational principles underlying movement generation, such as optimal feedback control, have often provided robust behavioral insights, but failed to connect these to their corresponding neural implementations. The basic relationship between cortical activity and the corresponding movement thus remains poorly understood. This understanding can allow us to uncover the underlying neural computations and dynamics required to perform diverse movements in a dynamic environment.

The representational perspective posits that cortical activity relates to abstract movement representations, such as movement direction or muscle features [1–11]. In recent years, this representational perspective has been challenged by the dynamical systems perspective, which states that motor cortical activity mainly contributes to supporting intrinsic dynamical features, and exhibits representational tuning only incidentally [12–17]. Moreover, goal-driven models trained to perform a motor

task considered analogous to that performed by the biological population of neurons have been successful at explaining population level dynamics [18–24]. However, these models lack biological and physical realism: they do not act on the real-time sensory feedback, do not interact with complex musculoskeletal dynamics, do not coordinate with intricate physical laws of the environment, are often trained on a very small subset of the features that the biological MC may consider as an input or output, and are not based on optimality principles. Therefore, these models often fail to generalize to unseen movements. For example, once a movement is learnt, we have the ability to produce this movement arbitrarily fast or slow, within a range, while maintaining accuracy. As a result, these models cannot be used for prediction of cortical activity during these unseen conditions. Building on previous studies [21–23, 25], we show that building in the biological realism of the musculoskeletal dynamics and incorporating the notion of optimality [26] in goal-driven models of MC can help better capture neural dynamics underlying movement generation while vastly improving generalization.

Here, we instantiate the point of view that the motor pathways in the brain act as a controller for the musculoskeletal system and drive optimal behavior under appropriate constraints [27–36]. Optimal feedback control theory has been successful at generating computational insights about the underlying control laws and providing behavioral-level predictions [37–52]. However, these models lack flexible neural network implementations of the controller, and fail to generate neural level predictions. A recent approach called MotorNet [53] includes training controllers with differentiable biomechanical models; however, the correspondence of the resulting controllers to the MC or generalization to unseen conditions has yet not been established. Moreover, neural constraints that arise from the evolutionary standpoint cannot be implemented using these models, such as minimization of neural firing rates.

Recently, deep reinforcement learning (DRL) has emerged as a promising way to train controllers for musculoskeletal models in highly complex tasks [54–58]. Using DRL, it is possible to train these models directly on high-dimensional sensory inputs to produce muscle activations that drive musculoskeletal models to produce complex movements, such as quick turning and walk-to-stand transitions [59]. The resulting behavioral-level dynamics have been shown to resemble experimentally recorded data [59].

In this research, we validate the neurophysiological plausibility of models obtained using DRL and propose a computational framework to generate behavioral- as well as population- and single-unit-level neural predictions for unobserved movements. We build in biological realism in this computational framework by basing it on the biological sensorimotor loop, incorporating anatomically accurate musculoskeletal models. We train these models using DRL as it shares the same notion of optimality (see Methods) as the optimal feedback control framework [60]. We also incorporate physical realism by simulating the resulting movements in highly advanced physics simulation engines. Moreover, for the first time, we show that the realistic neural-network-based implementation of a controller enables the implementation of neural constraints, such as minimization of neural firing rates, for movement generation.

This realism, along with the notion of optimality under suitable neural constraints, thus enables this framework to better capture and infer the behavior as well the underlying neural dynamics and single-unit firing rates. This framework also captures the computational principles and properties of motor control, such as generalization to novel movement conditions. Further, our results indicate that we can decode single-unit firing rates of MC using the activity of trained models even during unseen conditions. Thus, allowing these models to be used for hypothesis generation for prediction and analysis of neural activity during novel limb movements. We show that the developed framework is flexible enough to incorporate other known properties of MC, such as sensorimotor delays and forward models. This framework can thus significantly reduce the gap between the computational principles of movement generation and their corresponding population- and single-unit level neuronal implementation.

2 Results

2.1 Development of biologically and physically accurate goal-driven sensorimotor framework

We develop a goal-driven computational framework, μSim, to obtain a model of the MC with biological structural and functional properties (Fig. 1a). μSim consists of various independent, interactive and interchangeable modules (See Github page https://github.com/saxenalab-neuro/muSim). The

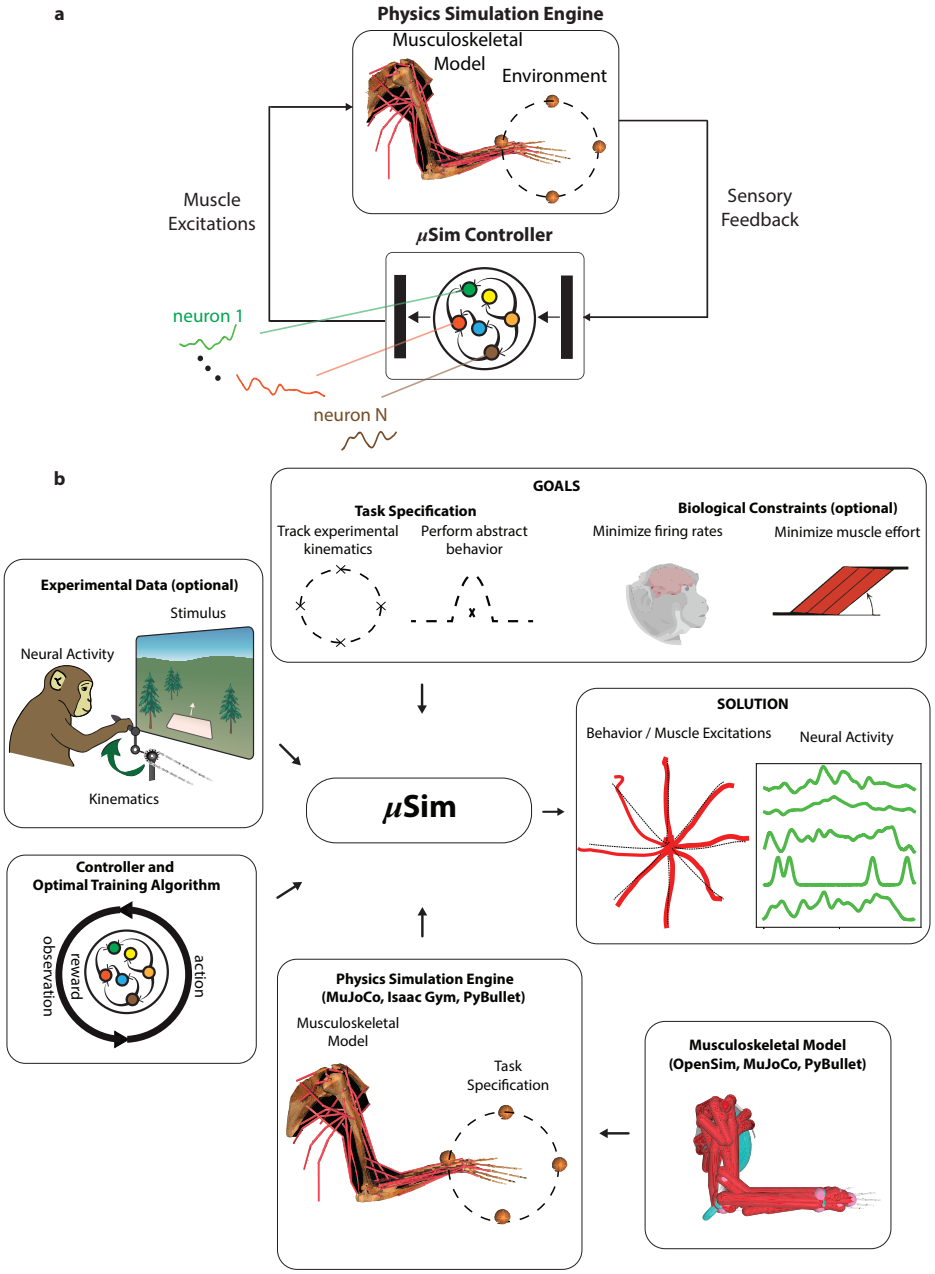


Fig. 1 Development and overview of the proposed goal-driven sensorimotor framework. a. In μSim, we build a biologically accurate model of the sensorimotor feedback loop. It consists of an anatomically accurate musculoskeletal model that can interact with its environment in a physics simulation engine. The physics simulation engine receives muscle excitations, executes the movement and generates the resulting sensory feedback. To execute a desired movement, the transformation from sensory feedback to muscle excitations is represented using a neural network controller. Here, the μSim controller consists of an RNN layer representing the model of MC and two feedforward layers representing the processing upstream and downstream of MC. **b.** The proposed goal-driven sensorimotor framework consists of various independent interactive input modules: musculoskeletal model, physics-simulation engine, controller and optimal training algorithm, experimental data, and goal specification. The biologically accurate joint/muscle - based musculoskeletal model can be defined in widely used and well-maintained softwares, such as OpenSim, MuJoCo or PyBullet. The defined musculoskeletal model may require conversion for use in inter-compatible physics-based simulation engines, such as Isaac Gym, MuJoCo or PyBullet. The physics simulation engine additionally requires environmental and task specifications, such as experimental kinematics or conditions. The training algorithm is used for training the weights of the μSim controller to transform sensory feedback into muscle excitations that optimally achieve the specified goals. The experimental data, such as kinematics, neural-activity or stimulus, can be used for goal-specification to train the controller to reproduce recorded movements under physical experimental conditions. The goal specification consists of two sub-modules: task specification and optional biological constraints. The task specification can be based on experimental conditions, such as track recorded kinematics, or abstract behavior, such as obstacle avoidance. Additionally, biological constraints, such as reproduction of recorded neural activity on observed conditions or minimization of neural firing rates, can also be implemented. As output, μSim then produces the muscle excitations and the controller's activity required to achieve specified goals.

modularity and flexibility allows the use of various musculoskeletal models, physics simulation engines and training algorithms with minimal modifications (Fig. 1b and Supplementary Fig. 1). μSim can thus be easily adapted for diverse use-cases, such as simulating realistic environmental conditions, building neural and behavioral constraints, and using musculoskeletal models of different animal species. Here, we explore the framework for a validated macaque limb model with 38 muscles [61], that was adapted from OpenSim to MuJoCo as in [62, 63] for computational efficiency.

μSim mimics the biological sensorimotor loop where the controller, representing the model of the MC, receives real-time sensory feedback and delivers muscle excitations to drive the musculoskeletal model producing diverse movements in the physics simulation environment (Fig. 1a). μSim consists of independent and interactive controller, environment, musculoskeletal model and goals specification modules (Fig. 1b). μSim supports the individual development and modification of these modules to incorporate biological and physical realism into the simulations. We base the controller on recurrent neural networks (RNNs) to mimic the recurrent connections of the biological MC. The controller consists of three layers, an RNN layer representing the model of MC, along with two feedforward layers representing the modular structure and the processing downstream and upstream of MC. The sensory feedback consists of visual and proprioceptive feedback, specifically muscle excitations, joint positions, joint velocities, hand and target positions and velocities and any high-level task input to the MC. We use anatomically accurate musculoskeletal models that receive muscle excitations to simulate the resulting movement. The physics simulation engine enables the implementation of diverse experimental conditions, such as contact / friction forces between bodies and ground reaction forces.

We use DRL to train the μSim controller to produce experimentally recorded movements under specified biological constraints (see Methods) (Fig. 1b). DRL trains the μSim controller to maximize a given reward function under specified neural- and kinematics- level constraints and is based on the same notion of optimality as optimal feedback control. In our approach, the reward function is designed such that the muscle excitations resulting in specified goals receive the maximum reward. The networks associated with reward function learning may be considered equivalent to other areas of the brain's reward system, such as basal ganglia, and may reflect dopaminergic projections to MC [64]. Here, we used the soft actor-critic (SAC) algorithm in a maximum entropy framework to train the controller (see Methods).

2.2 Trained μSim controller achieves high kinematic accuracy

Using our computational framework, μSim, we detail out the procedure for one specific set of experiments where an adult male rhesus macaque monkey was trained to produce a cyclic movement of the forelimb at 8 different speeds, while kinematics and single-unit activity from premotor and motor cortex were recorded [22]. We used experimentally recorded kinematics during a subset of four different speeds as a task specification for μSim to train the controller, leaving out other speed conditions for testing purposes. It is important to note that the recorded neural data was not used at all during μSim training.

While several μSim controllers were trained with high kinematic accuracy, here we analyzed one example model for comparison with the recorded kinematics and neural data. The model achieved a high kinematic accuracy on all the four training speeds and we observed a significantly low mean squared error (MSE) between the experimental and simulated trajectories (Fig. 2a).

2.3 MC population dynamics and single-unit activity can be decoded from the trained μSim controller's activity

We analyzed how well the trained μSim controller was able to capture the neural population dynamics of experimentally recorded MC neurons in the experimental conditions used for its training. We used canonical correlation analysis (CCA) for assessing correlations on the level of neural population dynamics and linear regression analysis (LRA) for assessing single-unit level decoding accuracy (see Methods).

CCA showed a high R^2 between the μSim RNN activity and the experimental MC population activity for the conditions that the μSim was trained on (Fig. 2c). Next, we trained a linear regressor from the μSim RNN's hidden activity to the recorded MC firing rates using data from all speed conditions except the held-out speed, and tested the linear model on the held-out condition using LRA (see Methods). The 'seen' speed is used for μSim training, but is held-out for LRA for finding the single-unit encoding fits below.

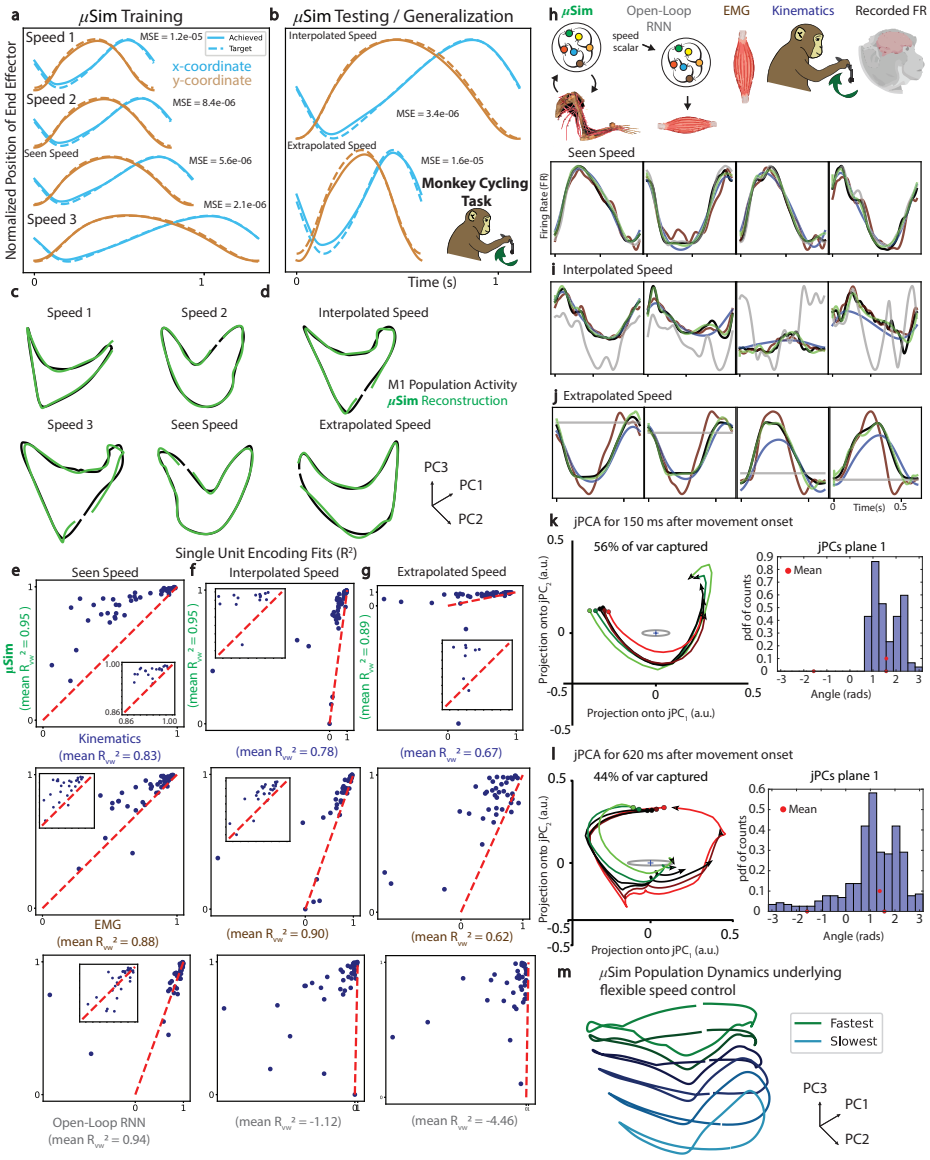


Fig. 2 The trained μSim controller achieves high kinematic accuracy and enables the decoding of neural dynamics and single MC units on training as well as unseen conditions for the monkey cycling task. **a**. The kinematics (x and y coordinates of hand) for the cycling movements produced by the trained μSim controller for the four training speed conditions are superimposed on corresponding experimental kinematics. The MSE values averaged across x and y coordinates for the training speed conditions are also shown (see Methods). **b.** Same as **a** but for the unseen interpolated and extrapolated speed conditions. **c.** The first 3 principal components (PCs) of MC population activity reconstructed from the μSim RNN activity using the inverse canonical correlation analysis (CCA) for the four training speed conditions are superimposed on the corresponding top 3 PCs of experimental MC activity. We observed a mean R^2 value of 1.00 between the actual and reconstructed MC population activity PCs for all speed conditions. **d.** Same as **c** but for the unseen testing speed conditions. We observed a R^2 value of 1.00 for both speed conditions. **e**. Scatter plots show the comparison of LRA-based single neuron decoding accuracy (R^2_{vw}) of the μSim against the existing goal-driven and representational models of MC on the seen speed. For each neuron (blue-dot), the R^2 of μSim RNN is shown along the vertical axis and R^2 of alternative models is shown along the horizontal axis. If a neuron lies above the dashed unity line, it is better predicted by the μSim as compared to the alternative models. Only those neurons are shown for which the $R^2 \geq 0$ for either of the two competing models. The variance-weighted mean R^2_{vw} values between the experimentally recorded firing rates and their reconstructions are also shown. **f and g.** Same as **e** but for the unseen interpolated and extrapolated speed conditions, respectively. **h.** Experimentally recorded firing rates (FR) of four example MC neurons (black) and their reconstructions decoded from the μSim RNN activity (green) using LRA on the seen speed are compared against their reconstructions generated from the existing goal-driven and representational models of MC: open-loop RNN (gray), experimentally recorded EMG (brown) and kinematics (blue). **i and j.** Same as **h** but for the unseen interpolated and extrapolated speeds, respectively. **k.** jPCA projections for the μSim activity for training and unseen testing speed conditions. Each trace shows the evolution of the μSim state over 150 ms after movement onset. Histogram shows the distribution of angles between the μSim state and its derivative for all analyzed times and conditions. Pure oscillatory dynamics result in angles near $\frac{\pi}{2}$. **l.** Same as **k** but for 620 ms after movement onset. **m.** μSim activity projected onto top 3 PCs when PCA was applied collectively for all training and unseen testing speed conditions.

We compared the single neuron decoding accuracy (R^2) of the μSim RNN to the existing goal-driven and representational MC models on the seen speed using LRA (Fig. 2e). We considered the following commonly used goal-driven- and representational-based models for comparison [1–11, 21, 22]: 1) Goal-driven open-loop RNN trained to transform a scalar speed signal to corresponding experimental recordings of muscle signals (electromyography; EMG) to provide a comparison against existing goal-driven models of MC ; 2) Empirically recorded EMG to test if the trained controller provides a comparison against the representational view that the cortical activity represents muscle commands; 3) Experimentally recorded kinematics to test if the MC model outperforms the representational view that signals mirroring the variables relevant to the task can capture cortical activity. The μSim RNN outperforms all alternative models in decoding accuracy for the speed condition tested (Fig. 2e). The firing rates of 4 example MC neurons reconstructed with LRA using different models for the seen condition are shown in Fig. 2h.

2.4 μSim generalizes to unseen conditions

We test the generalization ability of the μSim controller on 'interpolated' and 'extrapolated' speed conditions unseen during μSim training. The interpolated speed condition lies in between the range of training speeds and the extrapolated speed condition lies well outside the training speeds range. μSim, without further training, generalizes well to both the unseen conditions (Fig. 2b). We observed significantly low MSE values between the experimental and simulated trajectories averaged across x and y coordinates for the interpolated and extrapolated speed conditions, respectively.

After validating that μSim achieves a high kinematic accuracy on unseen speed conditions, we tested how well it captures the experimental neural data during those conditions. We first perform CCA separately on the interpolated and extrapolated speeds to recover a lower-dimensional subspace of maximum correlations between recorded MC and μSim RNN activity. We then perform inverse CCA to transform the μSim RNN activity from the maximum correlations subspace into the experimentally recorded MC activity subspace and reconstruct its top 3 PCs (Fig. 2d). CCA shows that the population activity of μSim RNN perfectly captures the MC population activity.

We then compared the decoding accuracy R^2 of μSim against the traditional goal-driven and representational based models of MC for the unseen interpolated and extrapolated speed conditions. For single-unit decoding accuracy comparisons, we performed LRA separately for interpolated and extrapolated speeds with each speed treated as held-out speed for LRA. μSim outperforms the traditional models in explaining cortical neural activity (Fig. 2f and 2g). The firing rates of 4 example MC neurons reconstructed with LRA using different models for each held-out unseen condition are shown in Fig. 2i and 2j.

The difference in the relative decoding accuracy is even more pronounced for the unseen speeds as compared to the training speed conditions. These results support the dynamical systems view of the MC. Furthermore, μSim outperforms other simple goal-driven models confirming that the biological and physical realism plays an important role in explaining the cortical neural activity during limb movements.

MC population activity has been shown to exhibit strong rotational dynamics for various movements [20, 21]. We used jPCA (see Methods) to quantify the rotational dynamics in μSim population activity. jPCA revealed the presence of strong rotational dynamics in μSim population activity for all speed conditions with rotational dynamics explaining a significant variance of the μSim activity (Fig. 2k and 2l). The mean of distribution of angles between the μSim state and its derivative lies close to $\frac{\pi}{2}$, indicating the presence of a strong rotational component. Moreover, for the monkey cycling task, MC population activity exhibits stacked elliptical neural trajectories having low trajectory tangling and separated along a speed dimension to enable flexible control of speed [22]. We found that μSim employs strikingly similar neural strategies for this task with elliptical trajectories separated along a speed dimension (Fig. 2m). Therefore, we conclude that various features of neural population dynamics, like rotational dynamics and low trajectory tangling, can be considered a consequence of optimally transforming the sensory feedback into muscle excitations under neural and musculoskeletal constraints to achieve desired movements. The emergence of biological neural population dynamics and structure thus allows μSim to capture cortical properties, such as generalization to unseen speed conditions, and the underlying computational principles.

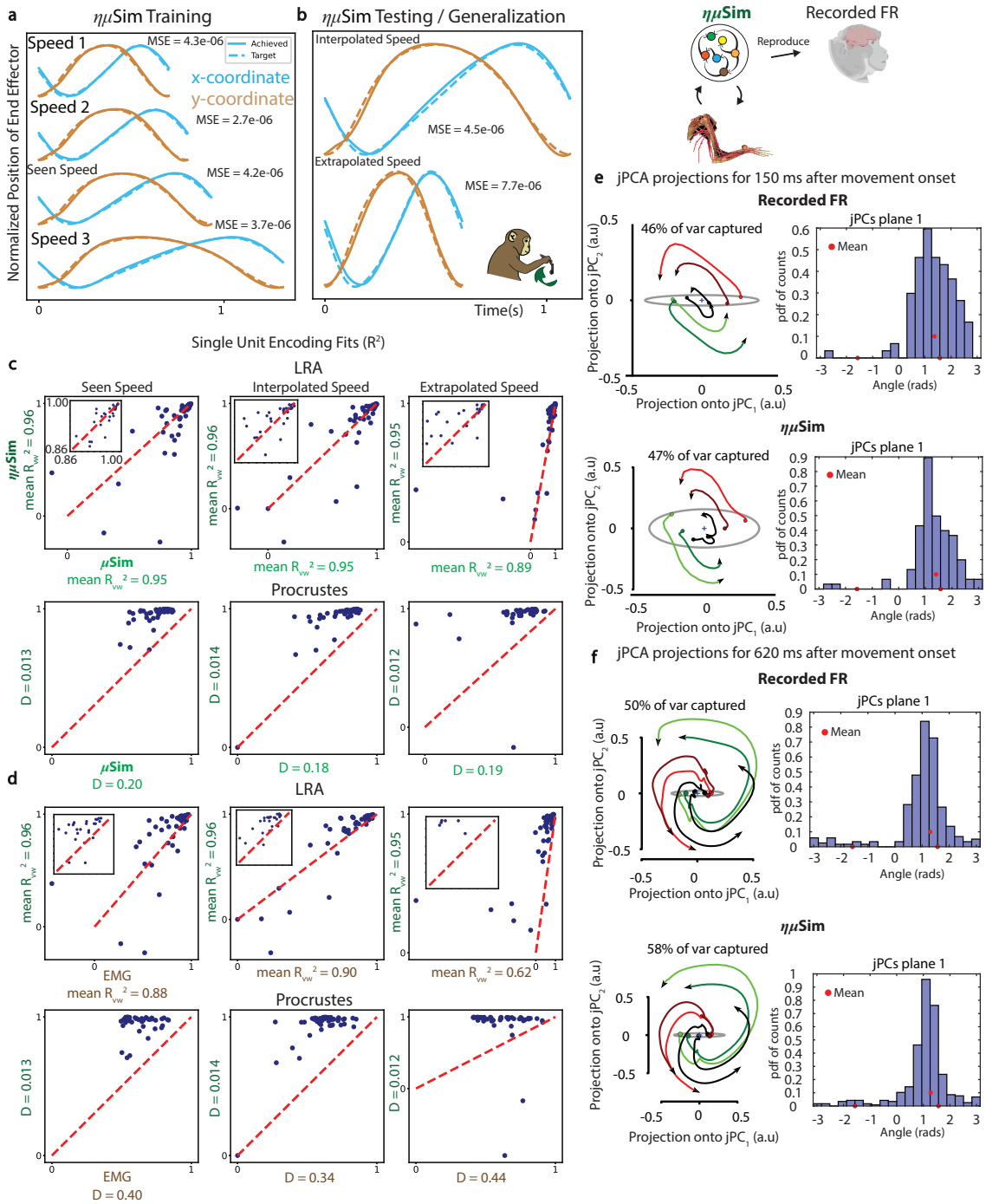


Fig. 3 Simultaneous goal- and data- driven modeling using $\eta\mu$Sim enables direct decoding of recorded single-unit activity in a generalizable manner. a. The kinematics of hand for the cycling movements produced by the trained $\eta\mu$Sim for the four training speed conditions are superimposed on corresponding experimental kinematics. The MSE values averaged across x and y coordinates for the training speed conditions are also shown (see Methods). **b.** Same as **a** but for the unseen interpolated and extrapolated speed conditions. **c.** Scatter plots show the comparison of single neuron decoding accuracy (R^2) of the $\eta\mu$Sim against the μSim on the seen, interpolated and extrapolated speeds using LRA (upper panel) and Procrustes (lower panel). The variance-weighted mean R^2_{vw} values between the experimentally recorded firing rates and their reconstructions using LRA are also shown (upper panel). Only those neurons are shown for which $R^2 \geq 0$ for either of the two competing models. D represents disparity for Procrustes (lower panel). **d.** Same as **c** but for comparison against EMG. **e.** jPCA projections of recorded MC data (upper panel) and $\eta\mu$Sim activity (lower panel) for 150 ms after movement onset for all training and testing speed conditions. Histogram shows the distribution of angle between the neural state and its derivative for all analyzed times and conditions. **f.** Same as **e** but for 620 ms after movement onset.

2.5 Simultaneous goal- and data- driven modeling enables generalizable single-unit level decoding on unobserved conditions

We showed that under appropriate constraints, μSim is able to predict single-unit and population-level neural activity using post-hoc analyses such as linear regression and CCA. Here, we constrain the solution space of μSim further to develop a direct correspondence with the recorded single units. The resulting constrained μSim is denoted as $\eta\mu$Sim. We develop a dynamical systems model of MC that is both goal- and data-driven: it transforms sensory feedback into muscle excitations that are converted to kinematics, while directly encoding recorded MC single units. During training, we constrain a subset of the $\eta\mu$Sim RNN's nodes to the experimentally observed MC single unit activity for training conditions (see Methods). We show that the $\eta\mu$Sim controller trained using DRL directly reproduces both the experimental MC single unit activity and the resulting kinematics (Fig. 3a). We then show that the trained model is generalizable to testing conditions that are significantly different from the training conditions (out-of-distribution generalization): it directly predicts the activity of the MC single units while simultaneously reproducing the behavior during those unseen conditions. The kinematics reproduced by $\eta\mu$Sim during these unseen conditions are shown in Fig. 3b.

To further interrogate the resulting structure of population dynamics in the trained networks, we use Procrustes Analysis (see Methods) to quantify how well the $\eta\mu$Sim RNN activity matches the experimentally recorded neural activity. Procrustes Analysis scales and rotates the network's activity to align it optimally with the experimentally recorded single unit activity. Procrustes Analysis is a well-established technique for single-unit experimental and network activities comparison and is a more stringent test of alignment than CCA and LRA [23, 65, 66].

We compare the single unit encoding accuracy of the trained $\eta\mu$Sim controller against the existing data- and goal- driven models of the MC: 1). Data-Driven Encoding EMG-based MC model trained to simply reconstruct experimental MC single unit activity from the corresponding muscle excitations (EMG). 2). Goal-driven neural-unconstrained μSim trained without enforcing the direct neural activity reproduction constraint, as in the preceding Section.

The scatter plots (Fig. 3c and 3d) compare the R^2 per neuron between its reconstruction produced by an alternative model (horizontal axis) against that produced by the $\eta\mu$Sim (vertical axis). We see that for all the seen and unseen speed conditions tested, most of the neurons lie above the dashed unity line, which means that they are being better predicted by the goal- and data- driven $\eta\mu$Sim model as compared to the alternative goal-driven neural-unconstrained μSim and data-driven EMG-based models. We also observe the lowest disparity D ($\mathcal{L}_2$ norm between the network reconstruction and recorded single-unit activity) for the $\eta\mu$Sim as compared to the alternative μSim and data-driven EMG-based models. Moreover, jPCA analysis reveals that the network dynamics of $\eta\mu$Sim mimic the MC neural population dynamics uncovering the underlying neural strategies for task representation (distinct initial conditions and trajectories) and movement generation (oscillatory dynamics) (Fig. 3e and 3f).

2.6 Modeling the contribution of sensory feedback and internal models on motor cortex activity dynamics and task execution

Here, we leverage the generalization ability of the $\eta\mu$Sim to explain and predict the computational principles underlying movement generation and their corresponding neural dynamics implementation. Little is known about how sensory feedback and internal forward models shape motor cortex (MC) dynamics during movement generation. We take a modeling perspective using the developed computational framework to probe this question. Goal-driven models of the MC based on recurrent neural networks (RNNs) when trained to transform high-level task-specific inputs into experimentally observed EMG exhibit rotational dynamics resembling the MC rotational activity patterns [20, 21]. Therefore, it is assumed that the intrinsic recurrent connections of MC give rise to its experimentally observed rotational dynamics with negligible contribution from sensory feedback. However, recent studies show that the models of MC based on feedforward networks without any recurrent connections also exhibit rotational activity patterns when trained to transform proprioceptive feedback into muscle activations, thus suggesting that sensory feedback contributes substantially to MC rotational dynamics [67]. We leverage the predictive ability of the $\eta\mu$Sim to evaluate these conflicting theories about the role of sensory feedback in MC dynamics.

To dissect the role of sensory inputs, we perform ablation studies by eliminating specific inputs to $\eta\mu$Sim and observe the effect on resulting movement kinematics and $\eta\mu$Sim's RNN activity patterns

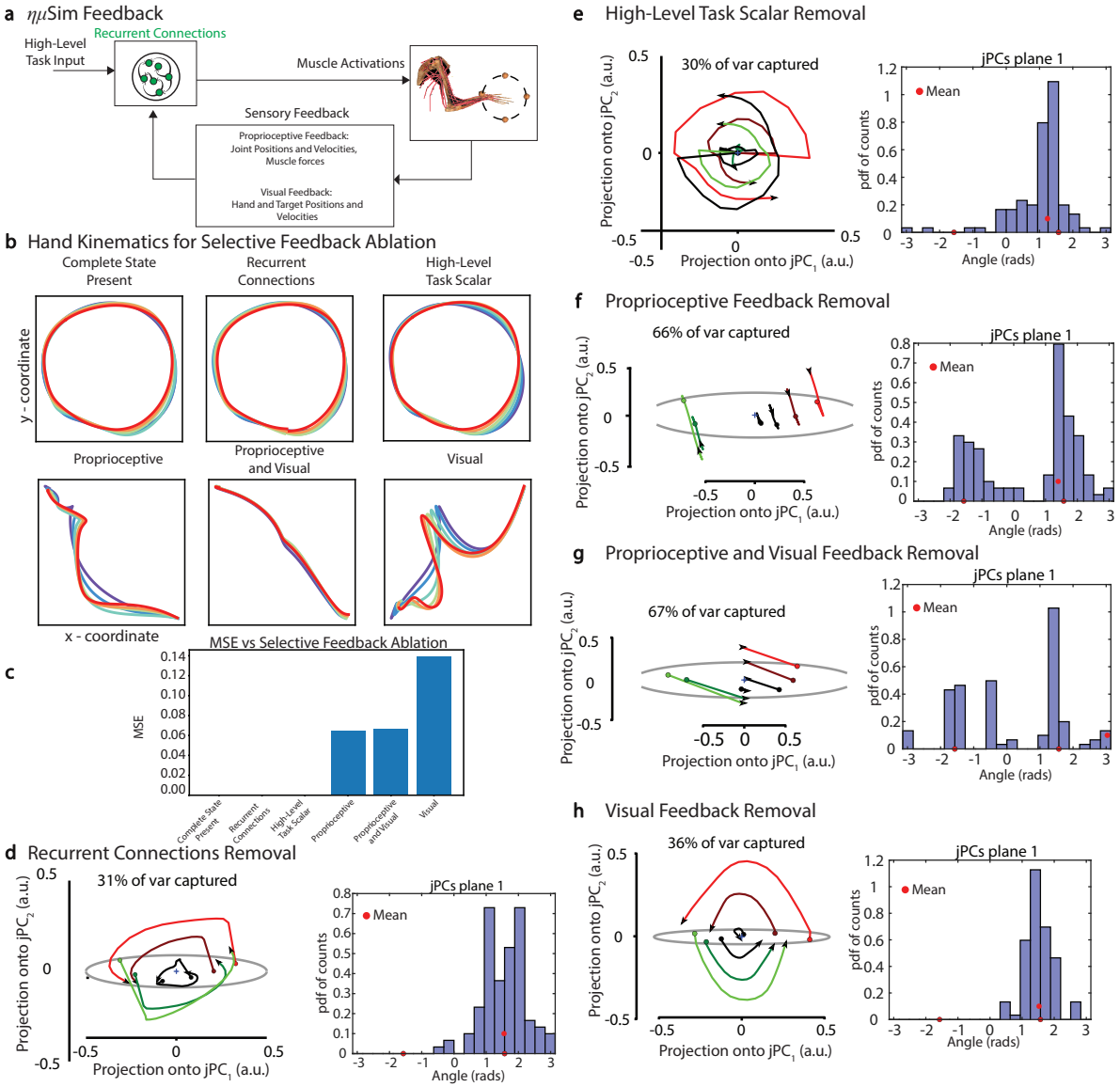


Fig. 4 **ημSim enables modeling the contribution of sensory feedback towards neural dynamics and task execution under optimality principles. a.** Overview of the sensorimotor loop modeled using ημSim emphasizing sensory feedback and recurrent connections. The high-level task input consists of a scalar signal representing a specific speed. Proprioceptive feedback consists of joint positions and velocities as well as muscle forces. Visual feedback consists of hand and target positions and velocities. Recurrent connections represent the recurrent weights of the RNN. **b.** Effect of various feedback ablation on movement execution. Different colors show the hand-trajectories (x and y coordinates) for different speed conditions. The title of each subplot shows the specific feedback that is ablated except the first subpanel for which complete state is present. **c.** MSE vs Selective feedback elimination. The x-axis represents the specific feedback that is ablated. The y-axis represents the MSE between the experimental and simulated kinematics summed over x and y coordinates for all the training and testing speed conditions. **d.** jPCA projections of ημSim state for 150 ms after movement onset for all training and testing speed conditions for recurrent connections ablation. Histogram shows the distribution of angle between the ημSim state and its derivative for all analyzed times and conditions. **e, f, g, and h.** Same as **d** but for high-level task scalar, proprioceptive, proprioceptive and visual, and visual feedback ablation, respectively.

(Fig. 4). We found that the recurrent connections' ablation significantly disrupted the initial rotational dynamics (Fig. 4d) and the variance explained by jPCs significantly reduced from 47% to 31%. To compensate for this disruption ημSim RNN employed angles greater than $\frac{\pi}{2}$. Eventually, ημSim is able to overcome this disruption and restore the rotational dynamics as the movement progresses (Supplementary Fig. 2a) and achieve the kinematics relatively well (Fig. 4b and 4c).

The ablation of the task-specific input significantly disrupts the rotational dynamics and separation of ημSim trajectories across different conditions (Fig. 4e). However, the network is able to overcome the disruption of rotational dynamics as the movement progresses (Supplementary Fig. 2b)

and achieve the task kinematics with negligible decrease in kinematics accuracy (Fig. 4b and 4c). The eventual emergence of condition-specific separation among trajectories (Supplementary Fig. 2b) indicates that $\eta\mu$Sim may be able to integrate the initially available sensory information to form task-specific internal forward models. The task-specific scalar input may thus reflect communication from the upstream brain regions representing the internal forward models that the network uses to separate the rotational trajectories across different conditions and execute the task relatively efficiently. Our results are consistent with previous studies suggesting the role of internal models in controlling the separation between neural trajectories [68].

Lastly, we examine the ablation of specific feedback to the model. If the proprioceptive feedback is ablated, $\eta\mu$Sim is not able to execute the task well (Fig. 4b and 4c). Surprisingly, the proprioceptive feedback ablation significantly increases the initial rotational dynamics (Fig. 4f). However, these rotational dynamics are significantly disrupted as the movement progresses (Supplementary Fig. 2c). If the proprioceptive and visual feedback is ablated, $\eta\mu$Sim is not able to execute the task at all (Fig. 4b and 4c). The $\eta\mu$Sim trajectories quickly settle at a fixed point (Fig. 4g) and the dynamics resemble pure translation/scaling as the mean of the distribution of angles approaches π. This correspondingly results in a linear movement of the musculoskeletal model's hand (Fig. 4b). Similarly, the visual feedback ablation significantly disrupts the task kinematics and rotational dynamics (Fig. 4b and 4h). Our results are consistent with previous computational approaches to study the role of proprioception during motor control [69].

As the original rotational dynamics are significantly disrupted by sensory feedback, including both the visual and proprioceptive feedback, high-level task scalar and the recurrent connections ablation, we conclude that these rotational dynamics are modulated synergistically by sensory brain regions, internal models and task-specific modules to achieve the desired movement. Therefore, rotational dynamics alone do not guarantee efficient task execution.

Biologically relevant sensorimotor delays dictate that MC acts on delayed feedback information. If the musculoskeletal simulator is in the current state s_t, the μSim RNN receives an outdated version of the state feedback, $s_{t-\Delta t}$ (Supplementary Fig. 3a). We observed that the μSim was able to achieve the delayed feedback task with high kinematic accuracy for $\Delta t = 60$ms [70]. We also observed a high R^2 between the μSim RNN activations at time t and the non-delayed state s_t. This may stem from the μSim RNN developing a representation of the state s_t from the delayed state $s_{t-\Delta t}$ in order to solve the task efficiently (Supplementary Fig. 3b).

2.7 μSim flexibly models diverse tasks and different animal species

Finally, we tested the modularity and flexibility of μSim for seamless modeling of diverse tasks and animal species. First, we tested the ability of μSim to capture neural dynamics across different tasks. For this purpose, we trained μSim on a center-out reaching task with 8 straight reach trajectories to the outer targets (see Methods) [71]. The controller achieves a high kinematic accuracy on all the experimentally observed conditions for the reaching task (Fig. 5a).

CCA revealed a high correlation between the trained μSim activity and the experimental MC population activity across all the reaching conditions (Fig. 5b and Supplementary Fig. 4a). LRA resulted in a high R^2 between μSim activity and single-unit activity for the 8 conditions, indicating that the controller was able to capture the firing rates of experimentally recorded MC single units. We then compared the LRA-based single neuron decoding accuracy (R^2) of μSim with the kinematics-based model of the MC (Fig. 5c and Supplementary Fig. 4b). We observed that μSim significantly outperformed the kinematics-based model of MC. Four example neurons reconstructed with LRA using different models are shown in Fig. 5d and Supplementary Fig. 4c.

Does the kinematic accuracy of the network determine the neural accuracy? We used Procrustes Analysis to quantify the role of the μSim's kinematic accuracy in enabling neural decoding. We trained several μSim controllers with low to high MSE values between the simulated and experimental kinematic trajectories averaged across the reaching conditions. We observed that the controllers with relatively lower MSE (higher kinematic accuracy averaged across the 8 reaching conditions) had lower disparity between their activity and the experimental MC firing rates (Fig. 5e). This shows that the kinematic accuracy of the trained controller plays an important role in decoding the experimental MC neurons.

Next, we tested if a trained μSim is able to generalize across tasks (out-of-distribution generalization). For this purpose, we compared the kinematic accuracy of three different controllers on the reaching task: 1) The μSim trained and tested on the monkey reaching task, Reaching μSim 2) The

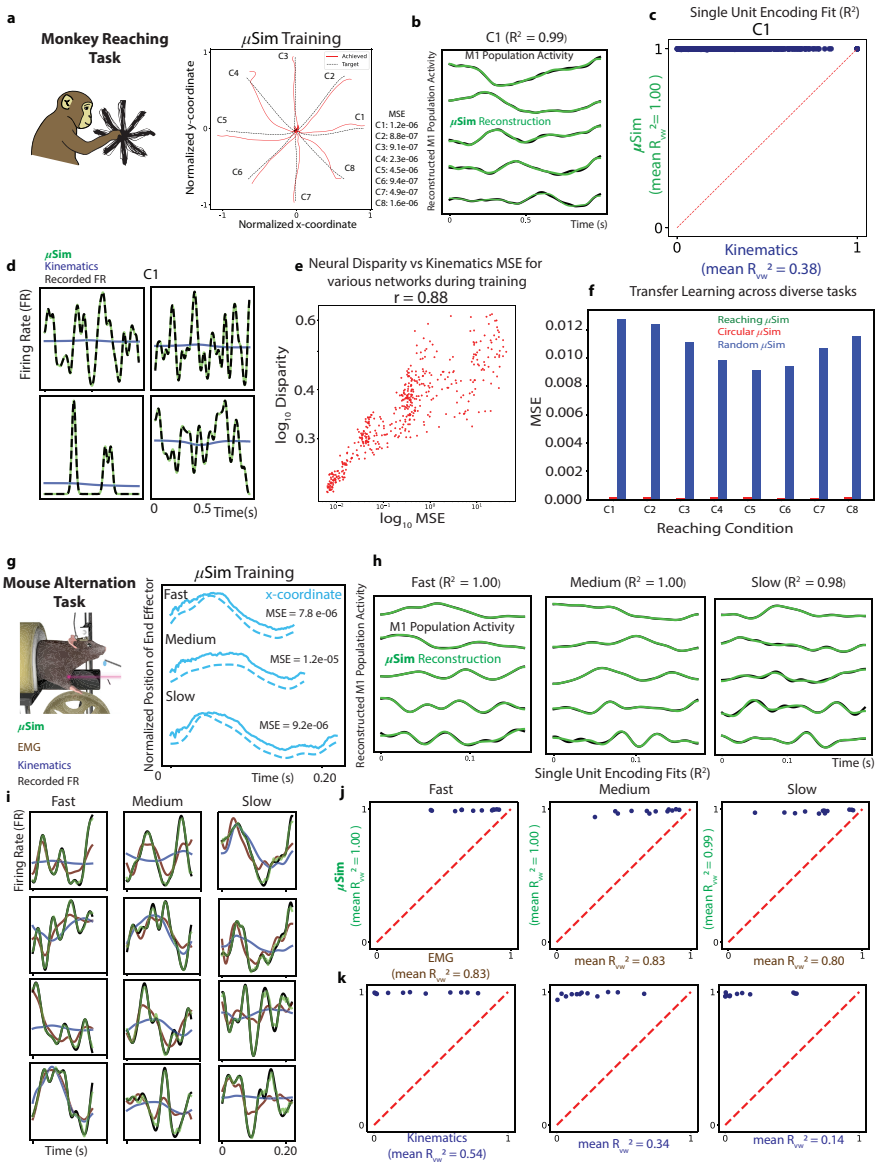


Fig. 5 The μSim framework flexibly models diverse tasks and animal species. a. μSim achieves high kinematic accuracy for a monkey reaching task. μSim simulated kinematics (red) are superimposed on the experimental kinematics (black) for all the experimentally observed reaching conditions, C1 through C8. The MSE between the μSim simulated kinematics and experimental kinematics averaged across x and y coordinates are also given. **b.** The top 5 PCs of MC population activity reconstructed from the trained μSim activity using the inverse canonical correlation analysis (CCA) are superimposed on the corresponding PCs of experimental MC activity for the C1. The mean R^2 values between the actual and reconstructed PCs are also shown. **c.** Scatter plot shows the comparison of LRA-based single neuron decoding accuracy (R^2) of the μSim against the kinematics-based model of MC for the C1. Variance weighted mean R^2_{vw} is also shown. **d.** Experimentally recorded firing rates of four example MC neurons (black) and their reconstructions decoded from the μSim activity (green) using the LRA on C1 are compared against their reconstructions generated from the kinematics-based model of MC (blue). **e.** The controller that achieves higher kinematic accuracy (lower MSE between the simulated and experimental trajectories) averaged across the 8 experimental reaching conditions shows lower disparity between its activity and the experimentally recorded MC activity using Procrustes Analysis. Pearson correlation coefficient r is also shown. **f.** Kinematic accuracies of Reaching μSim, Circular μSim and Random μSim on the reaching task. y-axis represents the MSE between the network's simulated kinematics and experimental kinematics averaged across the 8 reaching conditions. **g.** μSim achieves high kinematic accuracy for a mouse performing an alternation task. μSim simulated kinematics (solid) are superimposed on the experimental kinematics (dotted) for all the experimental conditions. MSE between the μSim and experimental kinematics is also shown. **h.** Same as **b** but for the mouse alternation task. **i.** Experimentally recorded firing rates of four example MC neurons (black) and their reconstructions decoded from the μSim activity (green) using the LRA on 3 experimental conditions are compared against their reconstructions generated from the EMG (brown) and kinematics (blue) -based model of the MC. **j.** Scatter plots show the comparison of LRA-based single neuron decoding accuracy (R^2) of μSim against the EMG-based model of MC for the 3 experimental conditions. Variance-weighted mean R^2_{vw} is also shown. **k.** Same as **j** but for the comparison against the kinematics-based model of MC.

μSim trained to perform the cycling task and tested on the reaching task, Cycling μSim. 3) A μSim controller initialized with random weights, Random μSim. The Cycling μSim significantly outperformed the Random μSim on the reaching task (Fig. 5f). We observed a relatively small difference between the kinematic accuracy of the Cycling μSim and Reaching μSim on the reaching task. This shows that the controllers obtained using this computational framework are able to generalize across diverse tasks.

Lastly, we trained μSim on a mouse alternation task with experimental single-unit MC activity, kinematics and EMG as in [72] (see Methods). We used an anatomically accurate musculoskeletal model of the mouse (see Methods) [73]. The μSim trained on the experimental kinematics achieved a high kinematic accuracy for the three training conditions (Fig. 5g). CCA revealed significantly high correlations between the top 5 PCs of experimentally recorded MC activity and the μSim RNN activity for the three conditions (Fig. 5h). We then used LRA to assess correlations on the level of single-unit activity. LRA resulted in high correlations between the μSim activity and experimental single-unit activity. Four example neurons reconstructed with LRA using different models are shown in Fig. 5i for the three conditions. LRA-based single neuron decoding accuracy (R^2) comparison of μSim with the EMG- and kinematics- based model of the MC showed that μSim significantly outperformed the alternative models (Fig. 5j and 5k). Therefore, μSim can be used in a seamless way across well-researched animal species in neuroscience.

2.8 Discussion

The brain is evolved to optimally interact with the physical environment under neural and musculoskeletal constraints. Here, we develop an anatomically and physically accurate computational framework, μSim, based on the biological sensorimotor loop, to obtain a model of MC. Under specified neural constraints, we use DRL to train the models of MC to optimally achieve diverse movements. The trained controller is able to capture and predict the neural dynamics and single-unit activity underlying movement generation in a generalizable manner. Constraining a subset of the μSim RNN's nodes to the experimental data during training conditions further enables it to develop one-to-one correspondence with the experimentally recorded single-units of MC and enables their direct prediction during unseen conditions. The resulting models capture the experimentally recorded neural population dynamics and structure surprisingly well. These models may capture higher-cognitive properties such as generalization to unseen conditions, even to those that are significantly different from training. We finally show that the developed framework accurately captures the computational principles governing the integration of sensory feedback, task specification, and recurrent neural dynamics underlying movement generation. Therefore, this computational framework can be used for prediction and hypothesis generation on both the behavioral and neural level. Future directions include the investigation of network remapping during the learning of new behaviors [74, 75].

μSim consists of various independent and interactive modules, such as the musculoskeletal model, physics simulation engine, controller and optimal training algorithm, experimental data and goals specification. These modules can be used and enhanced independently by machine learning engineers, computer scientists, control theorists, roboticists, biomedical engineers, medical professionals and neuroscientists to enable diverse applications. μSim can enhance our understanding of motor control on both behavioral and neural level, and uncover the role of sensory, cognitive and reward regions of the brain in movement generation. μSim will enable significant applications, such as enabling enhanced real-time decoding and encoding models for brain computer interfaces, discovering muscle excitations or designing stimulation signals to achieve various movements, building predictive simulations to improve rehabilitation and assistive devices research, discovering the effect of neural stimulation on resulting movement in deep brain stimulation, and building biologically plausibile control and machine learning models.

2.9 Acknowledgements

This work was supported by NIH Brain Initiative grant 1RF1DA056377-01. We are very grateful to the following researchers for making their experimental data available: Abigail Russo and Mark Churchland for the monkey cycling dataset, the Slutzky laboratory for the monkey reaching dataset, and Claire Warriner and Andrew Miri for the mouse alternation dataset.

References

[1] Edward V Evarts. "Relation of pyramidal tract activity to force exerted during voluntary movement." In: *Journal of neurophysiology* 31.1 (1968), pp. 14–27.

[2] Ferdinando A Mussa-Ivaldi. "Do neurons in the motor cortex encode movement direction? An alternative hypothesis". In: *Neuroscience letters* 91.1 (1988), pp. 106–111.

[3] Terence D Sanger. "Theoretical considerations for the analysis of population coding in motor cortex". In: *Neural Computation* 6.1 (1994), pp. 29–37.

[4] Emanuel Todorov. "Direct cortical control of muscle activation in voluntary arm movements: a model". In: *Nature neuroscience* 3.4 (2000), pp. 391–398.

[5] Nicholas G Hatsopoulos. "Encoding in the motor cortex: was evarts right after all? Focus on "motor cortex neural correlates of output kinematics and kinetics during isometric-force and arm-reaching tasks"". In: *Journal of neurophysiology* 94.4 (2005), pp. 2261–2262.

[6] Stephen H Scott. "Inconvenient truths about neural processing in primary motor cortex". In: *The Journal of physiology* 586.5 (2008), pp. 1217–1224.

[7] Tyson N Aflalo and Michael SA Graziano. "Relationship between unconstrained arm movements and single-neuron firing in the macaque motor cortex". In: *Journal of Neuroscience* 27.11 (2007), pp. 2760–2780.

[8] John F Kalaska. "From intention to action: motor cortex and the control of reaching movements". In: *Progress in motor control: a multidisciplinary perspective* (2009), pp. 139–178.

[9] Apostolos P Georgopoulos, Andrew B Schwartz, and Ronald E Kettner. "Neuronal population coding of movement direction". In: *Science* 233.4771 (1986), pp. 1416–1419.

[10] Krishna V Shenoy, Maneesh Sahani, and Mark M Churchland. "Cortical control of arm movements: a dynamical systems perspective". In: *Annual review of neuroscience* 36 (2013), pp. 337–359.

[11] Eberhard E Fetz. "Are movement parameters recognizably coded in the activity of single neurons?" In: *Behavioral and brain sciences* 15.4 (1992), pp. 679–690.

[12] Eberhard E Fetz. "Cortical mechanisms controlling limb movement". In: *Current opinion in neurobiology* 3.6 (1993), pp. 932–939.

[13] Uri Rokni and Haim Sompolinsky. "How the brain generates movement". In: *Neural computation* 24.2 (2012), pp. 289–331.

[14] Timothy P Lillicrap and Stephen H Scott. "Preference distributions of primary motor cortex neurons reflect control solutions optimized for limb biomechanics". In: *Neuron* 77.1 (2013), pp. 168–179.

[15] Michelle M Morrow and Lee E Miller. "Prediction of muscle activity by populations of sequentially recorded primary motor cortex neurons". In: *Journal of neurophysiology* 89.4 (2003), pp. 2279–2288.

[16] Marc H Schieber and Gil Rivlis. "Partial reconstruction of muscle activity from a pruned network of diverse motor cortex neurons". In: *Journal of neurophysiology* 97.1 (2007), pp. 70–82.

[17] Robert Ajemian et al. "Assessing the function of motor cortex: single-neuron models of how neural response is modulated by limb biomechanics". In: *Neuron* 58.3 (2008), pp. 414–428.

[18] EJ Buys et al. "Selective facilitation of different hand muscles by single corticospinal neurones in the conscious monkey." In: *The Journal of physiology* 381.1 (1986), pp. 529–549.

[19] Mark M Churchland et al. "Cortical preparatory activity: representation of movement or first cog in a dynamical machine?" In: *Neuron* 68.3 (2010), pp. 387–400.

[20] Mark M Churchland et al. "Neural population dynamics during reaching". In: *Nature* 487.7405 (2012), pp. 51–56.

[21] David Sussillo et al. "A neural network that finds a naturalistic solution for the production of muscle activity". In: *Nature neuroscience* 18.7 (2015), pp. 1025–1033.

[22] Shreya Saxena et al. "Motor cortex activity across movement speeds is predicted by network-level strategies for generating muscle activity". In: *Elife* 11 (2022), e67620.

[23] Jonathan A Michaels et al. "A goal-driven modular neural network predicts parietofrontal neural dynamics during grasping". In: *Proceedings of the national academy of sciences* 117.50 (2020), pp. 32124–32135.

[24] Josh Merel et al. "Deep neuroethology of a virtual rodent". In: *arXiv preprint arXiv:1911.09451* (2019).

[25] Omri Barak. "Recurrent neural networks as versatile tools of neuroscience research". In: *Current opinion in neurobiology* 46 (2017), pp. 1–6.
[26] Emanuel Todorov and Michael I Jordan. "Optimal feedback control as a theory of motor coordination". In: *Nature neuroscience* 5.11 (2002), pp. 1226–1235.
[27] Tomohiko Takei et al. "Transient deactivation of dorsal premotor cortex or parietal area 5 impairs feedback control of the limb in macaques". In: *Current Biology* 31.7 (2021), pp. 1476–1487.
[28] Stephen H Scott. "Optimal feedback control and the neural basis of volitional motor control". In: *Nature Reviews Neuroscience* 5.7 (2004), pp. 532–545.
[29] Edward S Boyden et al. "Millisecond-timescale, genetically targeted optical control of neural activity". In: *Nature neuroscience* 8.9 (2005), pp. 1263–1268.
[30] Eiman Azim et al. "Skilled reaching relies on a V2a propriospinal internal copy circuit". In: *Nature* 508.7496 (2014), pp. 357–363.
[31] Mackenzie Weygandt Mathis, Alexander Mathis, and Naoshige Uchida. "Somatosensory cortex plays an essential role in forelimb motor adaptation in mice". In: *Neuron* 93.6 (2017), pp. 1493–1503.
[32] J Meyer-Lohmann et al. "Effects of dentate cooling on precentral unit activity following torque pulse injections into elbow movements". In: *Brain research* 94.2 (Aug. 1975), pp. 237–251. ISSN: 0006-8993. DOI: 10.1016/0006-8993(75)90059-1.
[33] Mohsen Omrani et al. "Distributed task-specific processing of somatosensory feedback for voluntary motor control". In: *Elife* 5 (2016), e13141.
[34] J Andrew Pruszynski et al. "Primary motor cortex underlies multi-joint integration for fast feedback control". In: *Nature* 478.7369 (2011), pp. 387–390.
[35] Luke Stuart Urban. *An Electrophysiological Study Of Voluntary Movement and Spinal Cord Injury*. California Institute of Technology, 2018.
[36] Josh Merel, Matthew Botvinick, and Greg Wayne. "Hierarchical motor control in mammals and machines". In: *Nature communications* 10.1 (2019), p. 5489.
[37] CK Chow and DH Jacobson. "Studies of human locomotion via optimal programming". In: *Mathematical Biosciences* 10.3-4 (1971), pp. 239–306.
[38] Herbert Hatze and Johan D Buys. "Energy-optimal controls in the mammalian neuromuscular system". In: *Biological cybernetics* 27.1 (1977), pp. 9–20.
[39] Frank C Anderson and Marcus G Pandy. "Dynamic optimization of human walking". In: *J. Biomech. Eng.* 123.5 (2001), pp. 381–390.
[40] Tamar Flash and Neville Hogan. "The coordination of arm movements: an experimentally confirmed mathematical model". In: *Journal of neuroscience* 5.7 (1985), pp. 1688–1703.
[41] MG Pandy, BA Garner, and FC Anderson. "Optimal control of non-ballistic muscular movements: a constraint-based performance criterion for rising from a chair". In: (1995).
[42] Emanuel Todorov and Michael I Jordan. "Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements". In: *Journal of Neurophysiology* 80.2 (1998), pp. 696–714.
[43] Yoji Uno, Mitsuo Kawato, and Rika Suzuki. "Formation and control of optimal trajectory in human multijoint arm movement". In: *Biological cybernetics* 61.2 (1989), pp. 89–101.
[44] Marcus G Pandy et al. "An optimal control model for maximum-height human jumping". In: *Journal of biomechanics* 23.12 (1990), pp. 1185–1198.
[45] Gerald E Loeb, WS Levine, and Jiping He. "Understanding sensorimotor feedback through optimal control". In: *Cold Spring Harbor symposia on quantitative biology*. Vol. 55. Cold Spring Harbor Laboratory Press. 1990, pp. 791–803.
[46] Bruce Richard Hoff. *A computational description of the organization of human reaching and prehension*. University of Southern California, 1992.
[47] Bruce Hoff and Michael A Arbib. "Models of trajectory formation and temporal interaction of reach and grasp". In: *Journal of motor behavior* 25.3 (1993), pp. 175–192.
[48] Arthur D Kuo. "An optimal control model for analyzing human postural balance". In: *IEEE transactions on biomedical engineering* 42.1 (1995), pp. 87–101.
[49] Yury P Shimansky, Tao Kang, and Jiping He. "A novel model of motor learning capable of developing an optimal movement control law online from scratch". In: *Biological Cybernetics* 90 (2004), pp. 133–145.

[50] Weiwei Li, Emanuel Todorov, and Xiuchuan Pan. "Hierarchical optimal control of redundant biomechanical systems". In: *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 2. IEEE. 2004, pp. 4618–4621.

[51] Vivek R Athalye et al. "Invariant neural dynamics drive commands to control different movements". In: *Current Biology* 33.14 (2023), pp. 2962–2976.

[52] Maryam M Shanechi, Amy L Orsborn, and Jose M Carmena. "Robust brain-machine interface design using optimal feedback control modeling and adaptive point process filtering". In: *PLoS computational biology* 12.4 (2016), e1004730.

[53] Olivier Codol et al. "MotorNet: a Python toolbox for controlling differentiable biomechanical effectors with artificial neural networks". In: *bioRxiv* (2023), pp. 2023–02.

[54] Emanuel Joos, Fabien Péan, and Orcun Goksel. "Reinforcement learning of musculoskeletal control from functional simulations". In: *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part III 23*. Springer. 2020, pp. 135–145.

[55] Berat Denizdurduran, Henry Markram, and Marc-Oliver Gewaltig. "Optimum trajectory learning in musculoskeletal systems with model predictive control and deep reinforcement learning". In: *Biological cybernetics* 116.5-6 (2022), pp. 711–726.

[56] Jiacheng Weng, Ehsan Hashemi, and Arash Arami. "Natural walking with musculoskeletal models using deep reinforcement learning". In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 4156–4162.

[57] Jessica Abreu, Douglas C Crowder, and Robert F Kirsch. "Deep reinforcement learning for control of time-varying musculoskeletal systems with high fatigability: a feasibility study". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 30 (2022), pp. 2613–2622.

[58] Seungmoon Song et al. "Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation". In: *Journal of neuroengineering and rehabilitation* 18 (2021), pp. 1–17.

[59] Łukasz Kidziński et al. "Learning to run challenge solutions: Adapting reinforcement learning methods for neuromusculoskeletal environments". In: *The NIPS'17 Competition: Building Intelligent Systems*. Springer. 2018, pp. 121–153.

[60] Dimitri Bertsekas. *Reinforcement learning and optimal control*. Athena Scientific, 2019.

[61] Sherwin S Chan and Daniel W Moran. "Computational model of a primate arm: from hand position to joint angles, joint torques and muscle forces". In: *Journal of neural engineering* 3.4 (2006), p. 327.

[62] Aleksi Ikkala and Perttu Hämäläinen. "Converting biomechanical models from opensim to Mujoco". In: *Converging Clinical and Engineering Research on Neurorehabilitation IV: Proceedings of the 5th International Conference on Neurorehabilitation (ICNR2020), October 13–16, 2020*. Springer. 2022, pp. 277–281.

[63] Huawei Wang et al. "MyoSim: Fast and physiologically realistic MuJoCo models for musculoskeletal and exoskeletal studies". In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 8104–8111.

[64] Andreas R Luft and Stefanie Schwarz. "Dopaminergic signals in primary motor cortex". In: *International Journal of Developmental Neuroscience* 27.5 (2009), pp. 415–421.

[65] Simon Kornblith et al. "Similarity of neural network representations revisited". In: *International conference on machine learning*. PMLR. 2019, pp. 3519–3529.

[66] Peter H Schönemann. "A generalized solution of the orthogonal procrustes problem". In: *Psychometrika* 31.1 (1966), pp. 1–10.

[67] Hari Teja Kalidindi et al. "Rotational dynamics in motor cortex are consistent with a feedback controller". In: *Elife* 10 (2021), e67256.

[68] Seth W Egger et al. "Internal models of sensorimotor integration regulate cortical dynamics". In: *Nature neuroscience* 22.11 (2019), pp. 1871–1882.

[69] Kai J Sandbrink et al. "Contrasting action and posture coding with hierarchical deep neural network models of proprioception". In: *Elife* 12 (2023), e81499.

[70] Wolfgang Kruse et al. "Temporal relation of population activity in visual areas MT/MST and in primary motor cortex during visually guided tracking movements". In: *Cerebral Cortex* 12.5 (2002), pp. 466–476.

[71] Robert D Flint et al. "Accurate decoding of reaching movements from field potentials in the absence of spikes". In: *Journal of neural engineering* 9.4 (2012), p. 046006.

[72] Claire L Warriner et al. "Motor cortical influence relies on task-specific activity covariation". In: *Cell reports* 40.13 (2022).
[73] Shravan Tata Ramalingasetty et al. "A whole-body musculoskeletal model of the mouse". In: *Ieee Access* 9 (2021), pp. 163861–163881.
[74] Patrick T Sadtler et al. "Neural constraints on learning". In: *Nature* 512.7515 (2014), pp. 423–426.
[75] Matthew D Golub et al. "Learning by neural reassociation". In: *Nature neuroscience* 21.4 (2018), pp. 607–616.
[76] CG HARTMAN and JR STRAUS. *WL: The anatomy of the rhesus monkey.* 1933.
[77] Daris Ray Swindler and Joseph Erwin. "Systematics, evolution, and anatomy". In: *(No Title)* (1986).
[78] Kirsten M Graham and Stephen H Scott. "Morphometry of Macaca mulatta forelimb. III. Moment arm of shoulder and elbow muscles". In: *Journal of morphology* 255.3 (2003), pp. 301–314.
[79] Raeed H Chowdhury, Joshua I Glaser, and Lee E Miller. "Area 2 of primary somatosensory cortex encodes kinematics of the whole arm". In: *Elife* 9 (2020), e48198.
[80] Scott L Delp et al. "OpenSim: open-source software to create and analyze dynamic simulations of movement". In: *IEEE transactions on biomedical engineering* 54.11 (2007), pp. 1940–1950.
[81] Matthew Millard et al. "Flexing computational muscle: modeling and simulation of musculotendon dynamics". In: *Journal of biomechanical engineering* 135.2 (2013), p. 021005.
[82] Emanuel Todorov, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ international conference on intelligent robots and systems.* IEEE. 2012, pp. 5026–5033.
[83] Erwin Coumans and Yunfei Bai. "Pybullet, a python module for physics simulation for games, robotics and machine learning". In: (2016).
[84] Felix E Zajac. "Muscle and tendon: properties, models, scaling, and application to biomechanics and motor control." In: *Critical reviews in biomedical engineering* 17.4 (1989), pp. 359–411.
[85] Xuanchen Xiang and Simon Foo. "Recent advances in deep reinforcement learning applications for solving partially observable markov decision processes (pomdp) problems: Part 1—fundamentals and applications in games, robotics and natural language processing". In: *Machine Learning and Knowledge Extraction* 3.3 (2021), pp. 554–581.
[86] Pengfei Zhu et al. "On improving deep reinforcement learning for pomdps". In: *arXiv preprint arXiv:1704.07978* (2017).
[87] Daniel M Wolpert and Zoubin Ghahramani. "Computational principles of movement neuroscience". In: *Nature neuroscience* 3.11 (2000), pp. 1212–1217.
[88] Tamar Flash and Neville Hogan. "The coordination of arm movements: an experimentally confirmed mathematical model". In: *Journal of neuroscience* 5.7 (1985), pp. 1688–1703.
[89] Yoji Uno, Mitsuo Kawato, and Rika Suzuki. "Formation and control of optimal trajectory in human multijoint arm movement". In: *Biological cybernetics* 61.2 (1989), pp. 89–101.
[90] Christopher M Harris and Daniel M Wolpert. "Signal-dependent noise determines motor planning". In: *Nature* 394.6695 (1998), pp. 780–784.
[91] Emanuel Todorov. "Optimality principles in sensorimotor control". In: *Nature neuroscience* 7.9 (2004), pp. 907–915.
[92] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).
[93] Nir Vulkan. "An economist's perspective on probability matching". In: *Journal of economic surveys* 14.1 (2000), pp. 101–118.
[94] Andreas R Luft and Stefanie Schwarz. "Dopaminergic signals in primary motor cortex". In: *International Journal of Developmental Neuroscience* 27.5 (2009), pp. 415–421.
[95] Jonas A Hosp et al. "Dopaminergic projections from midbrain to primary motor cortex mediate motor skill learning". In: *Journal of Neuroscience* 31.7 (2011), pp. 2481–2487.
[96] David Weenink. "Canonical correlation analysis". In: *Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam.* Vol. 25. University of Amsterdam Amsterdam. 2003, pp. 81–99.

3 Methods

3.1 Experimental Data

3.1.1 Monkey Cycling Data

Head-restrained rhesus macaque monkeys were trained to perform a cycling task while sitting in a customized chair. During the controlled experiments, monkeys manipulated a pedal-like device with their right-arm while the left arm was loosely restrained. The real time horizontal and vertical hand positions were recorded. The wrist movement was restrained and the cycles were driven mainly by the movement of the elbow and shoulder. This resulted in a highly stereotyped arm movement across cycles.

The monkeys rotated the pedal to track a moving target with respect to their virtual first-person location on the monitor in front of them. Juice reward was dispensed so long as they maintained their virtual position close to the displayed target.

Conventional single electrodes driven by a hydraulic microdrive were used to make sequential neural recordings from a broader range of sites in primary motor cortex and adjacent aspect of dorsal premotor cortex. For all further analyses, these recordings were treated together as a single motor cortex population. Neural signals were amplified, filtered and manually sorted using Blackrock Microsystems Digital Hub and 128-channel Neural Signal Processor. During cycling, nearly all of the isolations made were responsive and those with low signal-to-noise ratios or insufficient trial counts were discarded. Gaussian filtering (20ms SD) was then applied to filter the spikes of each recorded neuron to produce estimated firing rate, which was then trial averaged.

The trials were divided into 8 speed bins. Kinematics, EMG and neural data were then averaged across the trials in each speed bin. In each speed bin, a high degree of training ensured stereotypical behavior across trials. Adaptive alignment procedure was applied to correct remaining slight misalignments. Aberrant trials deviating from stereotypical behavior were discarded. Kinematics, neural and EMG were then averaged across the resulting trials in each speed after applying this adaptive alignment procedure. Further details about data preprocessing are given in [22].

3.1.2 Monkey Reaching Data

A rhesus monkey was trained to perform a center-out eight-target reaching task while grasping a two-link manipulandum. The outer-targets were spaced 45° intervals around a circle. The monkey had to reach the outer target and hold for a liquid reward. The neural data was recorded from proximal arm area of primary motor cortex and dorsal premotor cortex contralateral to the arm used for performing the task. Kinematics and neural data were then averaged across stereotypical trials from the same session.

Further details about behavior and neural data recording and preprocessing are given in [71].

3.1.3 Mouse Alternation Data

Alternation task elicited MC dependent flexor-extensor alternation. Mice were head-fixed and trained to perform alternation task for a water reward while muscle activity from the forelimb was recorded. Neural activity in the left caudal forelimb area was recorded after mice had been fully trained. Kinematics, EMG and neural data were then averaged across stereotypical trials from the same session.

Further details about behavior and neural data recording and preprocessing are given in [72].

3.2 Musculoskeletal Models

3.2.1 Musculoskeletal Model of a Monkey Limb

We build on the 3D musculoskeletal model of a macaque monkey arm developed in [61]. The developed model consists of seven degrees of freedom (DoF) that include extension and flexion of the elbow, 3D rotation about the shoulder joint, supination and pronation of the lower forelimb, adduction/abduction and flexion/extension of the wrist. The five segments represented in the model are the hand, the torso, the radial side of the lower arm, the ulnar side of the lower arm and the upper arm. Shoulder abduction and adduction is its rotation around the x–axis and its rotation around y-axis is defined as internal and external rotation. Rotation of the shoulder about the z-axis is shoulder flexion and extension. Elbow flexion is rotation about the z-axis. An intermediate x rotation is

used to define the off-axis pronation and supination axis. Rotation about the x-axis at the center of the wrist is wrist flexion and extension. It is followed by rotation around the z-axis which is wrist abduction and adduction.

The model also consists of 38 muscles. These muscles are based on the anatomical data obtained from cadaveric studies [76, 77]. The muscle properties such as muscle/tendon length and pennation angle were obtained from the literature [78].

The musculoskeletal dynamics used for developing the forward dynamic model can be represented by the following equation

$$M_{7\times7}(\Theta_{7\times1})\ddot{\Theta}_{7\times1} = R_{7\times38}(\Theta_{7\times1})F_{38\times1} + G_{7\times1}(\Theta_{7\times1}) + V_{7\times1}(\Theta_{7\times1}, \dot{\Theta}_{7\times1}) + E_{7\times1} \tag{1}$$

Where M represents the mass distribution of the system given the current joint angles Θ. $\dot{\Theta}$ and $\ddot{\Theta}$ represent the joint velocities and accelerations, respectively. R is the moment arm matrix. F is a vector of muscle forces. G, V and E describe the moment contributions of gravitational, internal and external forces, respectively. The model is designed to be scalable to a generic monkey arm given the monkey mass. Further details of the model are given in [79].

This model is adapted for use in OpenSim [80]. We first replaced the existing muscle model with a more biologically accurate and computationally stable Millard muscle model [81]. However, the forward simulations based on biologically accurate musculoskeletal models in OpenSim are computationally expensive. This makes the downstream learning of the controller extremely inefficient. To overcome this challenge of slow forward simulations, we first convert the monkey arm model from OpenSim to an equivalent model in MuJoCo physics simulation engine [62, 63, 82]. MuJoCo is a state-of-the-art physics-based joint-constrained physics simulation engine that can reach forward simulation speeds of more than 600 times those of OpenSim [62]. Therefore, we optimized the OpenSim monkey arm model for use in MuJoCo by approximating the musculo-tendon units: we minimized the squared difference of joint positions between the OpenSim and MuJoCo forward simulations over all training trajectories. This resulted in an anatomically accurate musculoskeletal model in MuJoCo that can facilitate the downstream learning of the controller by producing fast forward simulations.

3.2.2 Musculoskeletal Model of a Mouse

We utilized an anatomically accurate mouse musculoskeletal model developed in [73], and built in the PyBullet framework [83]. PyBullet is a fast, stable, and open source physics simulator with a python application programming interface (API). The authors of [73] developed a simulator-agnostic muscle library to be integrated with PyBullet for the purposes of this model. μSim controller interacts with the right forelimb of this model, directly activating its 18 muscles while the rest of the skeletal model is fixed in place. Forelimb muscle attachment points were determined based on embryonic studies of mice; due to lack of experimental data, mainly distal muscles were included. The model does not include proximal muscles that originate from the spinal segment.

We now list the DoF for each joint in the forelimb. For shoulder joints, which were modeled as a spherical joint, there are three rotational DoF: retraction-protraction (rotation about the transversal axis), abduction-adduction (coronal axis), and external-internal rotation (sagittal axis). For elbow joints there are two DoF: extension-flexion (transversal axis) and supination-pronation (sagittal axis). Wrist joints additionally contain two DoF: extension-flexion (transversal axis), and abduction-adduction (coronal axis). More details as well as the joint range-of-motion and limits can be found in [73].

The muscles themselves are of the Hill-type [84], with activation dynamics given by

$$\frac{du(t)}{dt} = \frac{a(t) - u(t)}{\tau_{\text{act}}} \tag{2}$$

where $u(t)$ are the activation dynamics, $a(t)$ is the muscle excitation, and τ_{act} is the time constant. The muscle excitation signal $a(t)$ is determined by μSim controller, directly controlling the muscle activation in order to produce the necessary movements. We chose a biologically realistic starting position of the forelimb based on the joint range-of-motion given in [73].

3.3 Kinematics MSE

The MSE between the kinematics produced by μSim and experimental kinematics is calculated as follows:

$$MSE = \sum_{n \in x,y} \sum_{t=0}^{T} \frac{[h_n(t) - h_n^*(t)]^2}{2T} \tag{3}$$

where x and y represent the hand's x and y coordinates, respectively. h^* and h represent experimental and μSim hand kinematics, respectively. T represents the total number of time points in the corresponding cycling speed trajectory.

3.4 Formulation of the Reinforcement Learning Problem

We formulate the task of controlling musculoskeletal models to perform different movements as a reinforcement learning problem. This formulation consists of a policy network, critic networks and an environment. Given the current state $s_t \in S$ of the environment, the policy network outputs the probability distribution, represented by $\pi_\theta(a|s)$, over the possible actions $a \in A$. θ denotes the parameters of the policy network implemented using RNNs. The environment can be implemented using different physics simulation engines. In this work, we use MuJoCo [82] and PyBullet [83] physics engines for implementing the environment. The environment consists of musculoskeletal models that can interact with other physical objects. Advanced physics engines allow the implementation of physically realistic simulations under different constraints such as contacts. At each timestep t, the environment receives an action a_t from the policy network. a_t represents the muscle excitation signal or motor command. Given a_t, the resulting movement is executed and the environment transitions from the current state s_t to next state s_{t+1}. s_{t+1} represents the physical effects of the motor command executed in our biomechanical simulation. The state transitions can be stochastic in μSim and the conditional probability of reaching the next state s_{t+1} is given by $p(s_{t+1}|s_t, a_t)$. The state transitions are governed by the underlying dynamics of the musculoskeletal system and the physics environment. These dynamics are defined by differential equations represented by F:

$$\dot{s} = F(s, a) \tag{4}$$

Given the probability of starting in some initial state s_0, the probability of realizing a trajectory $T = (s_0, a_0, ..., s_N)$ under the policy $\pi_\theta(a|s)$ is given by:

$$p_\theta(T) = \prod_{n=0}^{N-1} \pi_\theta(a_n|s_n) p(s_{n+1}|s_n, a_n) \tag{5}$$

In current work, we implement deterministic transitions of the environment. At each timestep t, a reward r_t is also generated to quantify the performance of the policy network.

3.5 Environment Design

The environment consists of anatomically accurate musculoskeletal models of different subjects that can interact with other objects while obeying physical laws and constraints, such as collisions, friction, torque, contacts and ground reaction forces. This allows the simulation of freely moving subjects that can interact with their environment. This framework can thus be used to investigate motor control for freely moving subjects. Motor control for freely moving subjects has not been explored well previously due to limitations in training algorithms and simulation frameworks.

In current work, the objective is to reproduce experimentally observed behavior. The musculoskeletal models of different species in this work are fixed in space while the limbs/extremities and thus the end-effector can move freely. The experimentally recorded kinematics are represented in the simulation using abstract objects known as targets. At each timestep t, the position of the target is updated to reflect the experimentally recorded position to be tracked by the end-effector in a 3D plane. The performance of the policy network at each timestep t can thus be quantified by the distance between the end-effector position and the target position. The policy network is then trained to output muscle commands such that the end-effector tracks target position at each timestep t, thus reproducing the experimentally recorded behavior.

The same framework can be used to model the task or abstract behavior in the absence of experimentally recorded kinematics using sparse metrics for quantifying the performance of the policy network. For example, in the absence of experimental kinematics, the reaching task can be achieved by fixing the abstract target at the desired final position.

3.6 States/Actions

The environment state $\mathrm{s}_e(t)$ is filtered to form the state feedback s_t for the policy network. s_t may consist of complete or partial information required to solve the control problem. If partial information about the environment state $\mathrm{s}_e(t)$ is available such that the state feedback is imperfect or partially observable, the control problem in the DRL framework can be formulated as a partially observable Markov decision process (POMDP). Various formulations of DRL, such as deep hierarchical RL, can be used to solve the control problem for POMDPs [85]. In previous work, a DRL formulation with RNN implementation of the policy network has also been shown to solve control problems with a partially observed state [86]. This also justifies the use of RNN implementation of the policy network in current work, in addition to the existence of strong recurrent connections in the motor pathways. This may also point to the fact that the motor pathways may be able to construct complete state from partially observed information through these recurrent connections. In a dynamic environment with freely moving subjects, often only partial state information is available. Therefore, this computational framework can be used to solve such complex tasks.

In this work, s_t represents the sensory feedback and consists of both the visual and proprioceptive feedback. Specifically, state feedback s_t to the policy network consists of muscle kinematics (muscle activations at the last timestep), joint angle and velocity for each joint, positions and velocities of the end-effector in 3D space, positions and velocities of the target in the 3D space, the difference vector representing the distance between the end-effector and the target, and a scalar representing different conditions within a task ('task information').

The action vector $a \in A$ represents the muscle excitations or the motor command that are then transformed into muscle activations. These activations actuate some DoF by applying motor torque. Physics simulation engines, such as MuJoCo, can also accurately model additional active and passive forces, including gravitational and contact forces, in addition to these actuated forces. Such additional forces are not modeled accurately using biomechanics engines, such as OpenSim. Therefore, such biomechanics engines can not be used to model neural dynamics accurately in the context of freely moving subjects interacting with each other or the dynamic environment. The developed computational framework is therefore expected to advance our understanding of motor control in such complex tasks.

3.7 Reward Function Design

The reward function $r(s_t, a_t, n_t)$ specifies the behavior of the policy network. Motor pathways drive optimal behavior under specific neural and behavioral constraints. Therefore, the reward function consists of two parts: task specification $T(s_t, a_t)$ and constraints specification $C(s_t, a_t, n_t)$.

$$r(s_t, a_t, n_t) = T(s_t, a_t) + C(s_t, a_t, n_t) \tag{6}$$

where, n_t represents neural parameters, such as synaptic weights or firing rates. In the developed computational framework, these neural parameters n_t correspond to the parameters θ and activity n_t^π of the policy network. To keep the notation concise, we assume $n_t \subset s_t$.

The generated rewards can be immediate, such as provided at each timestep t, or they can sparse, such as provided only at the last timestep. Freely moving subjects in dynamic environments often receive sparse rewards that can be modeled using the proposed computational framework.

It has been proposed that the central nervous system (CNS) achieves a specific task under various kinematics- and behavioral-level constraints. Previously, constraints based on the minimization of muscle effort [26, 87] or maximizing the smoothness of the end-effector trajectory and of the torque commands have been proposed [88, 89]. The models based on these constraints have been successful at reproducing empirical kinematic data. However, it is not clear how these are measured by the CNS. Other models assume noise in the motor command that scales with its magnitude [90]. The CNS aims to then minimize the end-point variance given this motor noise. However, is also not clear if such scaling is a consequence of neural constraints, such as neuronal noise. Therefore, it remains an open question how we approach this constraint for novel, unrehearsed movements [87].

In this work, we instead assume and validate the existence of neural constraints, such as minimization of neural firing rates, that are suitable from an energy minimization and evolutionary standpoint. These neural constraints can also be thought of giving rise to other previously proposed kinematics- and behavioral-level constraints. We also validate that such constraints generalize across novel and unrehearsed movements. To test specific hypotheses, this framework can be used to implement various behavioral- and neural- level constraints, such as the ones described above. After training, the

resulting behavior and network activity under specified constraints can then be analyzed or compared with experimental data to validate their existence.

For the purposes of this work, these neural constraints can be implemented using regularizations on the policy network and are described below. Here, we design the reward function as consisting purely of the task specification $T(s_t, a_t)$. In this work, the task specification is designed to make the end-effector reproduce the empirical kinematics by tracking the target at each timestep t:

$$r(s_t, a_t) = T(s_t, a_t) = w_d d_{target} \tag{7}$$

where $d_{target} = \sum_{n \in x,y,z} k^{-|h_{n,t} - h^*_{n,t}|}$, $h_{x,t}$, $h_{y,t}$ and $h_{z,t}$ represent the x, y and z coordinates of the end-effector's position and $h^*_{x,t}$, $h^*_{y,t}$ and $h^*_{z,t}$ represent the x, y and z coordinates of the target's position at timestep t. This reward function $r(s_t, a_t)$ produces the maximum reward r_t for the motor command that results in the minimum distance between the end-effector and the target for the given timestep t. r_t is designed to decay exponentially with the increasing distance between the hand and the target to accelerate the learning of the policy network parameters. We use $w_d = 5.0$.

3.8 Notion of Optimality

Optimality principles have been used in motor control to specify the control laws giving rise to the generated behavior and to theoretically explain why the motor system behaves as it does [91]. Given the dynamics of the motor system, musculoskeletal models and environment as in (4), these control laws are usually specified by the cost functions, $c(s_t, a_t)$, that can be considered as analogous to the reward functions $r(s_t, a_t)$. For example, the control laws that achieve the task as accurately as possible while minimizing the energy consumption are considered more suitable than those that do not follow either of these constraints. The constraints that result in controls laws under which the experimental behavior can be inferred provide insights into quantities that the motor system is trying to optimize. Such constraints, such as minimization of energy consumption, should thus help us generalize across different novel and unrehearsed movements. Usually such constraints are specified in kinematics space, such as maximizing the smoothness of end-effector trajectory. The realsitic neural network implementation of control laws provides a way to instead specify these constraints in the neural space, such as minimization of neural firing rates. We show that the experimental behavior and neural dynamics across diverse and novel movements can be inferred under such neural constraints together with the biological accuracy of the developed sensorimotor loop.

3.9 Soft Actor-Critic (SAC) Design

We adapt the SAC algorithm to train the policy and critic networks to achieve the desired task. SAC is an off-policy RL algorithm based on the maximum entropy framework [92]. This algorithm is chosen because of the complex environmental dynamics and computational complexity of the biomechanical simulations.

The reward r_t generated at each timestep t determines the behavior of the policy trained using this algorithm under the neural constraints. The sum of the rewards discounted by the factor γ defines the return R.

$$R = \sum_{t=0}^{T} \gamma^t r_t \tag{8}$$

Where T is the last timestep. The discount factor $\gamma < 1$ determines the relative importance of the future rewards relative to the earlier rewards. We set $\gamma = 0.99$ for training μSim.

The high dimensional action space and the complex environmental dynamics in biomechanical simulations make exploration quite inefficient using standard RL algorithms. Here, an entropy term $H(\pi_\theta(.|s))$ is incorporated in the return to achieve efficient exploration required for training.

$$H(\pi_\theta(\cdot|s)) = E_{a \sim \pi_\theta}[-\log(\pi_\theta(a|s))] \tag{9}$$

The entropy term determines the stochasticity of the trained policy and is used to achieve efficient exploration and robust convergence towards the global optimum given the high dimensional action space and complex environmental dynamics. Moreover, it has been shown to considerably improve learning speed and stability over the training algorithms that maximize the standard RL objective of maximizing only the return R. Probabilistic matching that has been used to explain human decision making can be considered the biological equivalent of the maximum entropy RL framework [93].

The objective is to learn a policy that maximizes the following soft return:

$$J(\theta) = \sum_{t=0}^{T} E_{(s_t,a_t)\sim p_{\pi_\theta}}[r(s_t, a_t) + \kappa H(\pi_\theta(\cdot|s_t))] \tag{10}$$

The temperature coefficient κ controls the stochasticity of the trained policy by determining the relative importance of reward against the entropy term. κ is adjusted automatically during training using the dual gradient-descent implemented in the SAC algorithm [92].

The SAC algorithm uses policy iteration to train the parameters of the policy network to maximize the objective function $J(\theta)$. To reduce the variance of the sampled trajectories during training, the SAC algorithm makes use of the critic network in addition to the policy network.

3.10 Critic Network

The critic network represents the transformation from the state-action pair (s_t, a_t) to its q-value $q(s_t, a_t)$ which in turn represents the expected soft-return of motor command a_t given the sensory feedback s_t in the maximum entropy RL framework [92]. The critic network parameterizes the q-value q_ϕ through parameters ϕ. In this work, the critic network consists of two RNN-layers with 256 nodes each.

Dopaminergic projections from the ventral tegmental area to the MC may reflect the neural correlates of the reward signals [94]. The reward regions of the brain, such as basal ganglia along with these dopaminergic projections, can be considered equivalent to the critic network. Disruption of these reward signals can inhibit further learning during forelimb reaching movements in rats, a prediction that can be obtained using the actor-critic architecture used [95].

3.11 Policy Network

The actor/policy network parameterizes the policy π_θ through the parameters θ. The policy network represents the mapping from state feedback s_t to the probability distribution over the muscle excitation space. In the brain itself, various regions are involved in the transformation from the sensory feedback to the motor command. The sensory feedback is first processed in the sensory and visual processing regions of the brain that share strong reciprocal connections with the premotor areas. The premotor cortex processes the feedback further and has strong reciprocal connections to the MC. The MC then transforms the processed feedback into motor commands through subcortical and spinal cord projections.

We base the architecture of the policy network to mimic the modular structure and reciprocal connections of the motor pathways involved in the movement generation. Therefore, the policy network consists of three layers representing the sensory and premotor/motor regions and the final subcortical and spinal cord projections. The layers representing the premotor and motor regions are based on RNNs to mimic the recurrent connections. For comparison with the recorded cortical data, we use the activity of the RNN layers representing the premotor and motor cortex.

The policy network consists of three layers. The first layer is a feedforward forward layer with the following input-output transformation:

$$u_i(t) = \sigma_1 \left[\sum_{k=1}^{I_1} W^U_{ik} s_k(t) + b^U_i \right] \tag{11}$$

where $s_k(t)$ is the input sensory feedback with dimensionality I_1. σ_1 is the non-linearity for the first layer. W^U and b^U represent the weights and biases for the first layer.

The second layer consists of an RNN. RNN can be considered a discrete dynamical system with the following dynamics:

$$x_i(t) = \sigma_2 \left[r_i(t) \right] \tag{12}$$

with

$$r_i(t) = \sum_{k=1}^{N_1} W^I_{ik} u_k(t) + b^I_i + \sum_{k=1}^{N_2} W^H_{ik} x_k(t-1) + b^H_i \tag{13}$$

Where r represents the inputs to the non-linear activation function σ_2 and x represents the corresponding output of RNN hidden layer. N_1 is the number of units in the first feedforward layer and N_2 is the dimensionality of the hidden layer of the RNN. σ_2 is the non-linearity for the RNN layer. W^I and b^I represent the input weights and biases, respectively. W^H and b^H represent the recurrent weights and biases for the RNN layer, respectively.

The final layer is a feedforward layer with the following input-output transformation:

$$z_i(t) = \sum_{k=1}^{N_2} W^Z_{ik} x_k(t) + b^Z_i \tag{14}$$

The output of the third layer, z, represents the muscle command with dimensionality N_3. W^Z and b^Z represent the weights and biases for the final feedforward layer, respectively.

3.12 Neural Constraints

Here, we hypothesize that the brain is evolved to produce optimal behavior under neural constraints. These neural constraints can be implemented as regularizations on the networks. Here, we implement three neural constraints as regularizations on the policy network activity.

The first regularization term is an L_2 penalty on the input and output weights of the three layers which encourages the minimization of synaptic weights or sparse connections between the network nodes.

$$R_{SW} = \sum_{i,k}^{N_1,I_1} \left(W^U_{ik}\right)^2 + \sum_{i,k}^{N_2,N_1} \left(W^H_{ik}\right)^2 + \sum_{i,k}^{N_3,N_2} \left(W^Z_{ik}\right)^2 \tag{15}$$

The second regularization term encourages the minimization of the neural firing rates. It also prevents the network activity from saturating:

$$R_{FR} = \sum_{i=1}^{N_1} \int_0^T u_i^2(t)dt + \sum_{i=1}^{N_2} \int_0^T x_i^2(t)dt + \sum_{i=1}^{N_3} \int_0^T z_i^2(t)dt \tag{16}$$

Where T consists of cumulative timepoints for all the conditions the network is trained on.

Finally, we implement a third regularization term that encourages the network to achieve the task while making simple trajectories in the low-dimensional state-space as proposed in [21].

$$R_{SD} = \int_0^T \frac{dW^H x(t)}{dr(t)} dt \tag{17}$$

In this work, we observed that R_{SD} does not play a significant role in increasing the correlation or similarity between the network and neural activities.

Therefore, the final loss function consists of the following terms:

$$L(\theta) = -J(\theta) + \alpha R_{SW} + \beta R_{FR} + \zeta R_{SD} \tag{18}$$

Where $\theta = W^U, W^H, W^I, W^Z, b^U, b^H, b^I, b^Z$ represents all the parameters of the policy network.

The policy network parameters are then trained using gradient descent to minimize the loss $L(\theta)$. We use Adam optimizer to update the parameters of the loss function. This loss function is used for μSim training.

We used $\alpha = 0.001$, $\beta = 0.01$ and $\zeta = 0.1$. We used $N_1 = N_2 = 256$. Additionally, we used $\sigma_1 = \sigma_2 = tanh$.

3.13 Simultaneous goal- and data- driven modeling

To implement the simultaneous goal- and data- driven modeling, we enforce an additional constraint on a subset of the RNN units to follow recorded neural activity for the training conditions. Specifically, we minimize the follow loss between the network and recorded single unit activity:

$$L_{GDM}(\theta) = \sum_{i=1}^{N_R} \sum_{c=1}^{C} \int_0^{T_C} [x_i(c,t) - n_i(c,t)]^2 dt \tag{19}$$

Where T_c represents the number of timesteps per condition, C is the total number of training conditions, n represents recorded neural activity, and N_R is the total number of recorded neurons.

At each timestep t, the simultaneous goal- and data- driven modeling loss minimizes the difference between the activities of the subset of RNN units and the corresponding recorded neurons. The final loss is:

$$L(\theta) = -J(\theta) + \alpha R_{SW} + \beta R_{FR} + \zeta R_{SD} + \tau L_{GDM}(\theta) \tag{20}$$

This loss function is used for $\eta\mu$Sim training.

We used $\tau = 10^4$ to account for relatively less number of experimentally recorded neurons as compared to total number of units in the policy network and relatively smaller magnitude of L_{GDM} as compared to other terms in (20).

3.14 CCA

We used CCA to find the correlations R^2 between the trained network and experimentally recorded neural population responses [96]. CCA finds the weightings for the units in network and experimental datasets such that the reweighted datasets are maximally correlated. Both the network and recorded neural activities were first reduced to ten dimensions using principal components analysis (PCA). CCA was then applied to recover the subspace of maximum correlations between the recorded and network activities during one complete cycle of the movement. Inverse CCA was applied to transform the network activities from the maximum correlations subspace back into the recorded activities subspace. We reported the reconstruction comparison and R^2 between the network and experimentally recorded activities in this subspace.

3.15 jPCA

We used jPCA [20] to quantify the oscillatory dynamics in neural state $x(t, c)$ across times, t, and conditions, c. jPCA provides summary features, such as quality of fit and variance explained, relevant to the hypothesis that the neural state evolves according to the oscillatory dynamics. It also allows the visualization of the two-dimensional projection of the neural data containing the oscillatory dynamics. As a preprocessing step, we first applied PCA to the neural data having dimensionality equal to the number of recorded neurons to reduce it to the top 4 PCs capturing $\geq 90\%$ variance. Using jPCA, we analyzed two different time periods of the neural activity: 1) 150ms after movement onset; 2) 620ms after movement onset.

3.16 Linear Regression Analysis

We used linear regression analysis (LRA) to compare the network and experimentally recorded neural single-unit level activities. For the monkey cycling task, we first fit a linear model with ridge regression on all the conditions except the held-out condition. The network and neural activities for training conditions excluding the held-out condition are first concatenated along the time dimension separately. Ridge regression is then used to fit a linear model in which the concatenated activity for each recorded neuron is determined by the concatenated network activity separately. For testing, we used this trained linear model to transform the network activity for each held out condition into the recorded neural activity subspace. The transformed network activity is then compared with the corresponding actual single-unit level neural activity for the held-out condition during the movement period to find the correlations. We used a similar procedure for comparing the kinematics and EMG models.

For the monkey reaching and the mouse alternation task, we fit a linear regressor using ridge regression separately on each condition. Ridge regression is used to fit a linear model in which the activity of each recorded neuron is determined by the network activity separately for the movement period. We used a similar procedure for comparing the kinematics and EMG models.

We used a regularization coefficient of 5×10^{-2} for LRA.

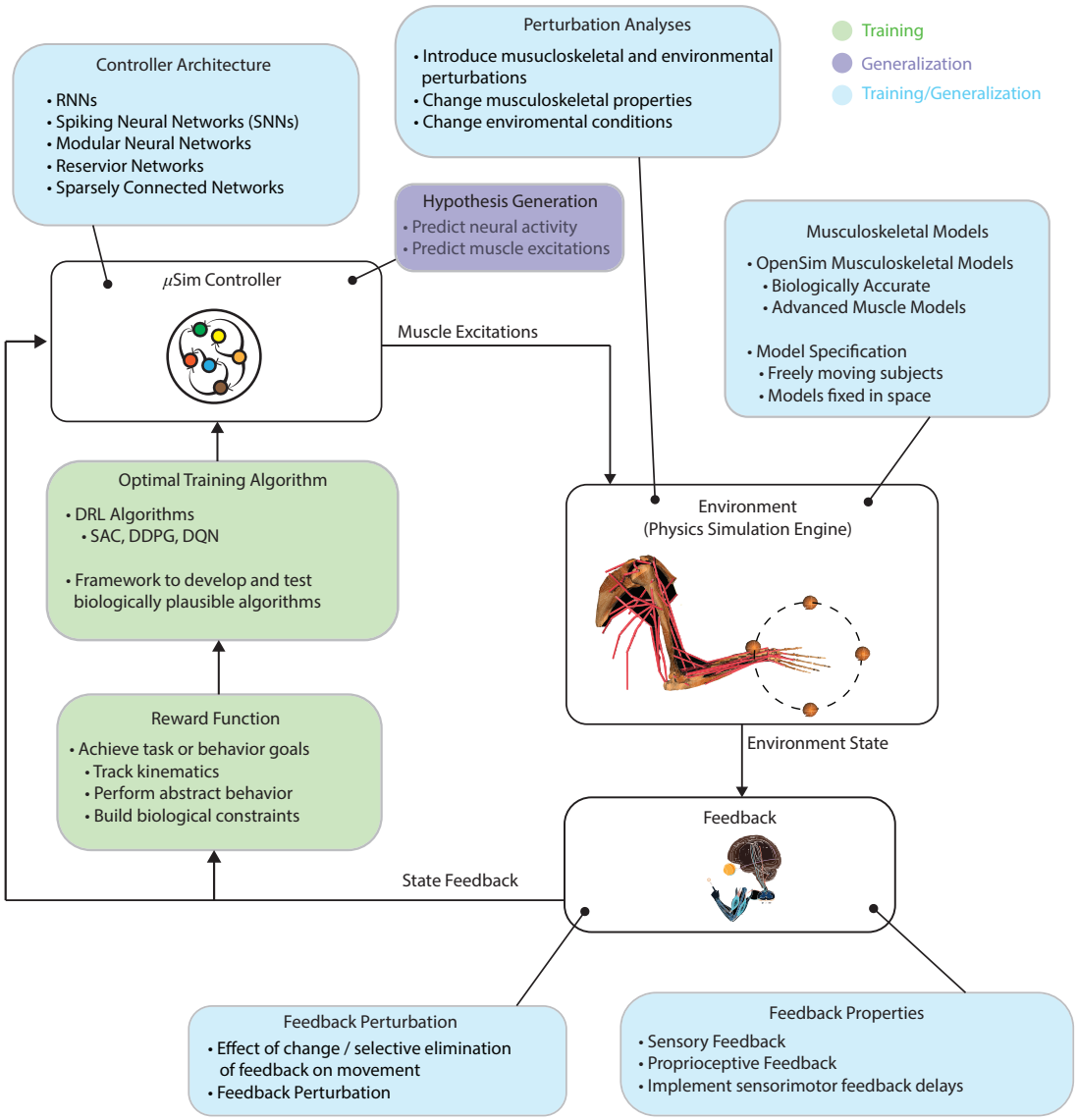
3.17 Procrustes

We used Procrustes Analysis to compare the network and neural single-unit level activities [65, 66]. Procrustes applies linear transformations, such as scaling and rotation, to the network activity to align it with the neural activity for the given condition. Procrustes minimizes the following loss:

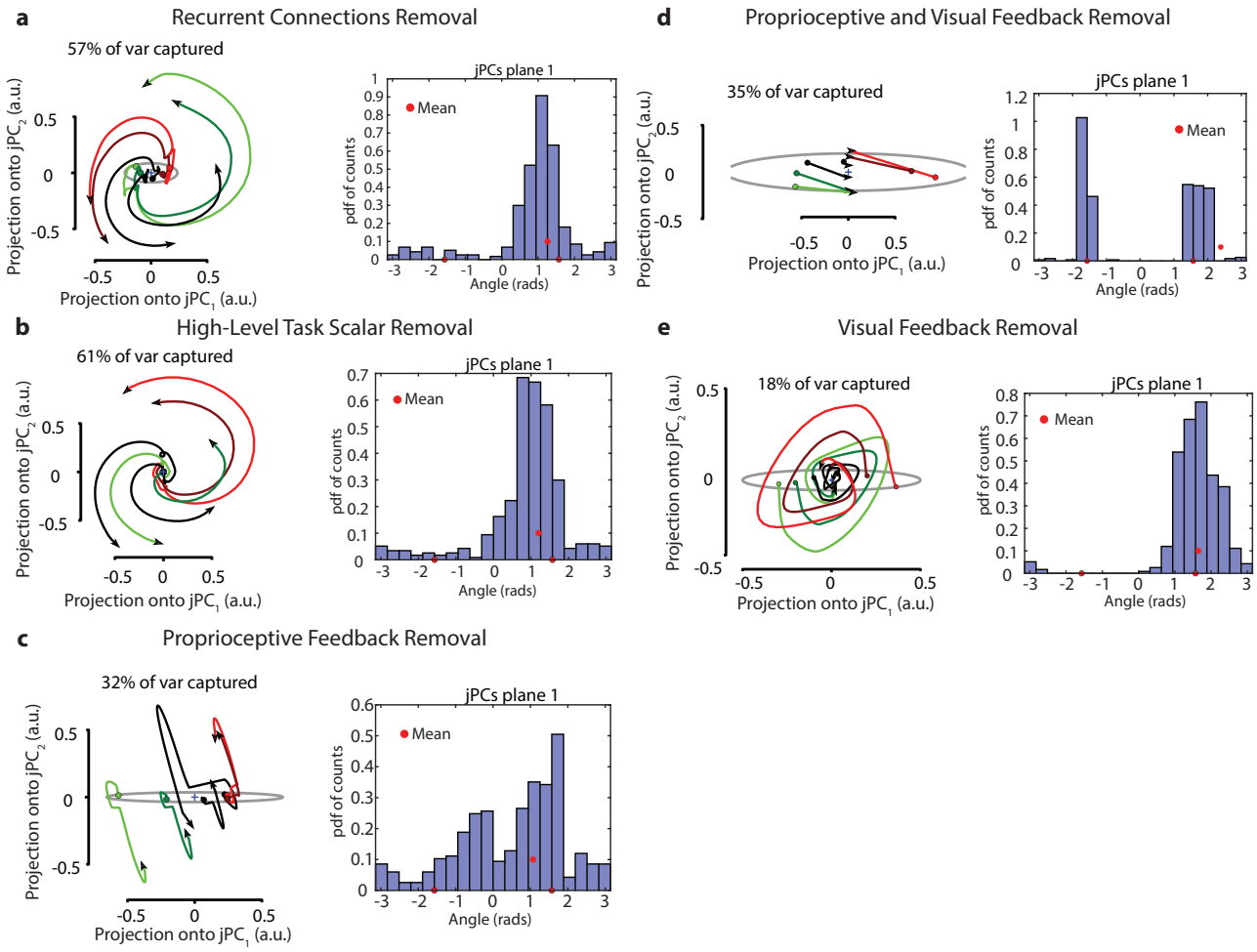
$$D = \sum (data_{net} - data_{neu})^2 \tag{21}$$

Where $data_{net}$ represents the network activity and $data_{neu}$ represents the recorded neural activity. The rows of the data matrix correspond to the timepoints and the columns correspond to the units/neurons. If the number of units in the neural data is less than the network activity, we first apply PCA to the network activity to make the number of units equal. Conversely, if the number of units in the network activity is less than the neural data, we first append the columns with zeros to make them equal. Here, D is also known as the disparity.

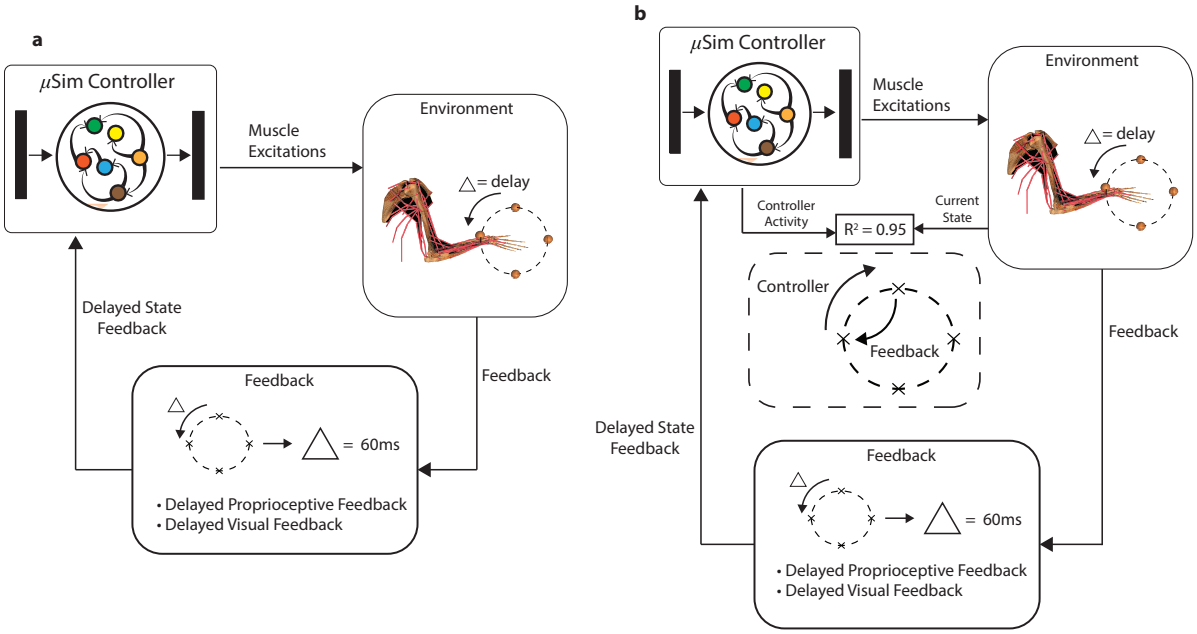
4 Supplementary Figures



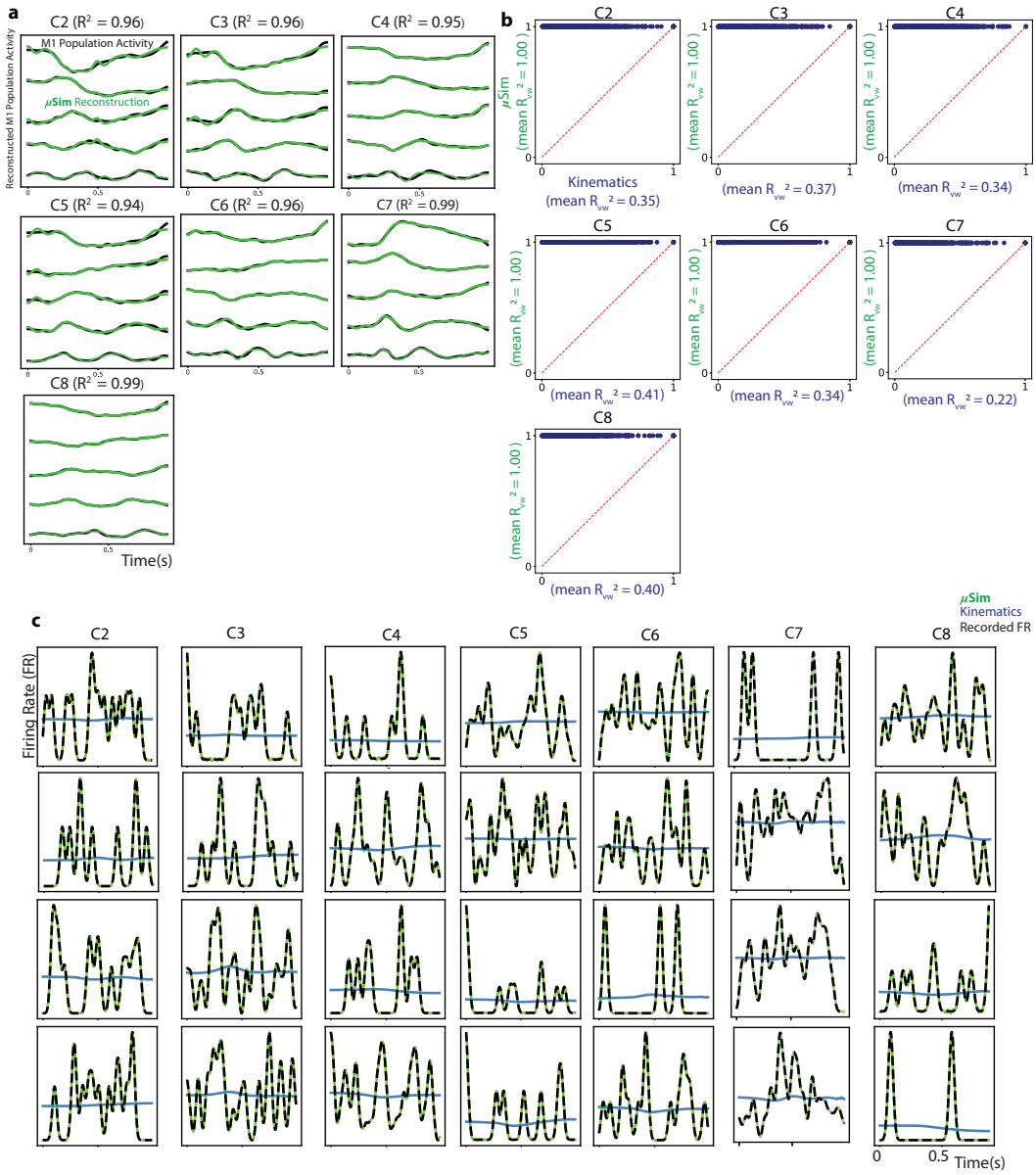
Supplementary Fig. 1 Overview of the training and generalization strategy. The interactive controller, environment and feedback modules can be independently enhanced to mimic the biological properties of sensorimotor loop and physical properties of experimental environment conditions. The controller can consist of various architectures to mimic the structural properties of biological neuronal populations. Similarly, the environment implemented in advanced physics simulation engines is based on biologically accurate musculoskeletal models and the underlying physics enables the implementation of various realistic conditions, such as freely moving subjects and contact forces. The state feedback models the real-time visual and proprioceptive feedback. During training, the controller is trained on a specified subset of experimental conditions using specific training algorithms, such as DRL. The reward function receives real-time feedback and generates relatively higher rewards for muscle excitations that achieve specified goals. The training algorithm then uses the generated rewards to train the controller's neural network using back propagation. During and after training, the controller's generalization ability can be tested on various environmental and feedback perturbation analyses. After training, if the controller generalizes well to the unseen experimental conditions, this property can be used for hypothesis generation and prediction of behavior and neural activity during experimentally unobserved conditions.



Supplementary Fig. 2 Effect of feedback ablation on neural dynamics under optimality principles. a. jPCA projections of $\eta\mu$Sim state for 620 ms after movement onset for all training and testing speed conditions for recurrent connections ablation. Histogram shows the distribution of angle between the $\eta\mu$Sim state and its derivative for all analyzed times and conditions. **b, c, d, and e.** Same as **a** but for high-level task scalar, proprioceptive, proprioceptive and visual, and visual feedback ablation, respectively.



Supplementary Fig. 3 **μSim is able to model the sensorimotor feedback delays. a.** Schematic showing the goal-driven sensorimotor framework, μSim, in the presence of feedback delays. The current musculoskeletal and environmental state is delayed by 60ms constituting the delayed sensory feedback. This delayed feedback is then fed to the μSim controller as input. **b.** For the μSim controller trained with delayed state feedback, we observe a high mean R^2 between the μSim controller's activity and current state feedback. The feedback is delayed by 60ms but the μSim develops a representation of the current state to solve the task. This indicates that the μSim was able to form a representation of current feedback from its delayed version.



Supplementary Fig. 4 **μSim flexibly models diverse tasks. a.** The top 5 PCs of MC population activity reconstructed from the trained μSim activity using the inverse canonical correlation analysis (CCA) are superimposed on the corresponding PCs of experimental MC activity for the C2-C8. The mean R^2 values between the actual and reconstructed PCs are also shown. **b.** Scatter plots show the comparison of LRA-based single neuron decoding accuracy (R^2) of the μSim against the kinematics-based model of MC for the C2-C8. Variance weighted mean R^2_{vw} is also shown. **c.** Experimentally recorded firing rates of four example MC neurons (black) and their reconstructions decoded from the μSim activity (green) using the LRA on C2-C8 are compared against their reconstructions generated from the kinematics-based model of MC (blue).