Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Scalable quasi-static self-modeling for physical legged locomotion

Yuhang Hu (yuhang.hu@columbia.edu)
Columbia University
Robert Kwiatkowski
Columbia University
Boyuan Chen
Duke University
Philippe Wyder
Columbia University
Hod Lipson
Columbia University

Article

Keywords:

Posted Date: July 7th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3121605/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Scalable quasi-static self-modeling for physical legged locomotion

Yuhang Hu[1], Robert Kwiatkowski[2], Boyuan Chen[2,*],
Philippe Martin Wyder[1], Hod Lipson[1,3]

[1]Department of Mechanical Engineering, Columbia University, New York, NY, USA
[2]Department of Computer Science, Columbia University, New York, NY, USA
[3]Data Science Institute, Columbia University, New York, NY, USA

Self-modeling refers to an agent's ability to learn a predictive model of its own behavior. A continuously adapted self-model can serve as an internal simulator, enabling the agent to plan and assess various potential behaviors internally, reducing the need for expensive physical experimentation. Self-models are especially important in legged locomotion, where manual modeling is difficult, reinforcement learning is slow, and physical experimentation is risky. Here, we propose a Quasi-static Self-Modeling framework that focuses on learning a predictive model only of high-level quasi-static dynamics, rather than a continuous model. Experimental results on a 12-degree-of-freedom-legged robot demonstrate improvements over model-free and traditional model-based continuous approaches. Using 80 diverse robot morphologies, we confirm a correlation of R^2=0.94 between the improvements rendered by our method and the DoF of the robot, suggesting that as future robots increase in complexity, this approach will become more valuable.

*Current address: Mechanical Engineering and Materials Science, Duke University, Durham, NC, USA

Introduction

Self-Modeling is a concept that spans different disciplines, including human cognition, animal behavior, and robotic systems (*1–3*). The term refers to the process where an agent, whether biological or artificial, learns to construct some internal representation of its own physical properties, encompassing various aspects such as morphology, kinematics, and dynamics (*4, 5*), as well as sensing, actuation, control and even planning.

Most robots today are trained, at least partially, inside some simulator, but these simulators are typically designed by a human engineers who decide what to model and how. In contrast, self-modeling involves an agent learning to simulate itself based on its ongoing experiences in the world. The advantage of an agent creating its own simulation include the ability for the simulator to keep up with changes through the lifetime of the agent, as well as the ability to create a simpler and more efficient simulator that captures only on aspects relevant to the agent's goals.

Previous studies showed the use of self-modeling in robotics to accomplish various tasks. For instance, Bongard et al. presented a self-modeling algorithm for a quadrupedal walking robot with an unconventional motor layout (*6*). However, this model relied on a basic simulator, which necessitated a dependency on known physics and mechanics, and therefore was not a fully data-driven approach. A more recent study of self-modeling successfully predicted outcomes without relying on any known physics, but was limited to just the end effector of a robot (*7*). Chen et al. proposed a fully-body visual self-modeling approach that learns the 3D shape and kinematics of the entire body of the robot as a statistical model, to achieve 3D motion planning and control (*3*). Additionally, Hu et al. introduced an egocentric visual self-modeling technique that allows the robot to utilize purely 2D visual perception for performing basic dynamic prediction for locomotion (*8*).

Deep reinforcement learning for robot locomotion

Reinforcement learning allows agents to automatically approximate ideal behavior based on interaction with the environment, while maximizing some notion of cumulative reward to achieve a goal (*9, 10*). Reinforcement learning (RL) has proven to be a powerful tool for controlling legged robots (*11–27*). However, the large amount of data needed to generate policies and the instability of those policies has posed substantial challenges to application of in real-world environments, especially in fields such as legged locomotion where collection of data is energetically costly, risky and slow (*28–32*).

Reinforcement learning that also learns a model of the world (Model Predictive Control, MPC) can be used to reduce the data requirement model-free reinforcement learning. An agent first learns a model of the world that can predict future states (*33–35*). The key advantage of model-based reinforcement learning is data efficiency. Compared to model-free reinforcement learning, it allows the agent to learn more about the environment from fewer interactions (*36*). However, model-based approaches face challenges regarding world model accuracy, particularly in complex environments.

Accurate learning of complex, high-dimensional systems is generally difficult, and inaccuracies in the model can lead to local optimal policies (*36–38*). Further, the need for planning in model-based methods can add to computational complexity, which is not suitable for real-time applications (*39, 40*). While many general-purpose algorithms have been proposed to address the inaccuracies inherent in the learned world models, applying model-based reinforcement learning specifically to legged robots in the real world remains relatively sparse (*41–45*).

One approach to reducing the challenges of model-based reinforcement learning is to decompose the world model into a self-model (which covers just the agent), and ”everything else”. As the tasks and environments change, the self remains relatively stationary. Therefore, learning a self-model rather than a world model, can be more tractable and scalable in the long term.

It can also allow task transfer and skill retention across a changing body. The key question we address in this paper is how to create an effective self model, especially for systems that involve complex kinematics, such as legged robots.

Robotic systems typically oscillate between static and dynamic phases - for instance, a legged robot transitioning between airborne and landing states during locomotion. This behavior informed the design of our Quasi-static Self-Modeling framework, which distinguishes itself from traditional model-based reinforcement learning.

A quasi-static model is designed to learn the dynamics and predict future states, focusing solely on predicting the changes between static waypoints. This is a marked contrast to conventional world models that aim to approximate a continuous model of the dynamics by predicting every interaction of an agent at small intervals, without regards to which parts of the motion are more salient than others. Our inspiration is the observation that while running, we humans likely spend more time predicting the placement of our next footstep than predicting our precise pose in mid flight in between footsteps.

Compared to model-free reinforcement learning, The data collected for QSM is used to predict the next quasi-static state without understanding the full underlying dynamics, thus significantly reducing the amount of data required. Our method thus improves sampling efficiency and stable performance. We also demonstrate a linear increase in utility with the robot's complexity, an advantage that is particularly important in training future robotic systems with higher degrees of freedom.

The primary contributions of our study can be summarized as follows:

1. We introduce a quasi-static self-modeling framework that offers improved sampling efficiency and stable performance in a robot locomotion task compared to model-free and model-based reinforcement learning.

2. We demonstrate a linear increase in the utility of quasi-static self-models with the com-

plexity of the robot, providing prediction of the the value of self modeling to future legged locomotion system.

Quasi-static self-models for robot locomotion

In this section, we detail our method of constructing a quasi-static self-model for the locomotion task. Traditional robotic self-models typically operate in an offline dataset paradigm where task-agnostic data is obtained through motor babbling. The primary goal of this approach is to forecast the entire dynamics and kinematics of the robot. This method proves successful for low-degree-of-freedom machines and fully-actuated robots as it effectively captures their relatively straightforward dynamics.

However, when applied to the underactuated, high degree of freedom legged robots, this approach faces substantial challenges. The inherent complexity of the dynamics and kinematics of legged robots makes constructing an accurate dynamic model via the traditional method challenging. The offline data collection strategy is based on motor babbling, which leads to different distributions of the training data set compared to experiences of the robot locomotion when applied in the context of locomotion. Motor babbling requires a large amount of data to train a model that can accurately predict the states during locomotion tasks (Fig. 2b the dashed line).

We introduce a novel self-modeling approach for robots that is more data efficient and excels in task-specific learning. The resulting self-model operates in an incremental learning framework that self-improves through exploration and exploitation (Fig. 1). The quasi-static self-model input consists of the robot's current state and action, as shown in Figure1. The action input is comprised of two elements: the motor babbling action (Ab) and the recommended action (Ar). The Motor Babbling action, a set of random actions, serves as the exploration. Meanwhile, the recommended action is a calculated action based on the highest predicted re-

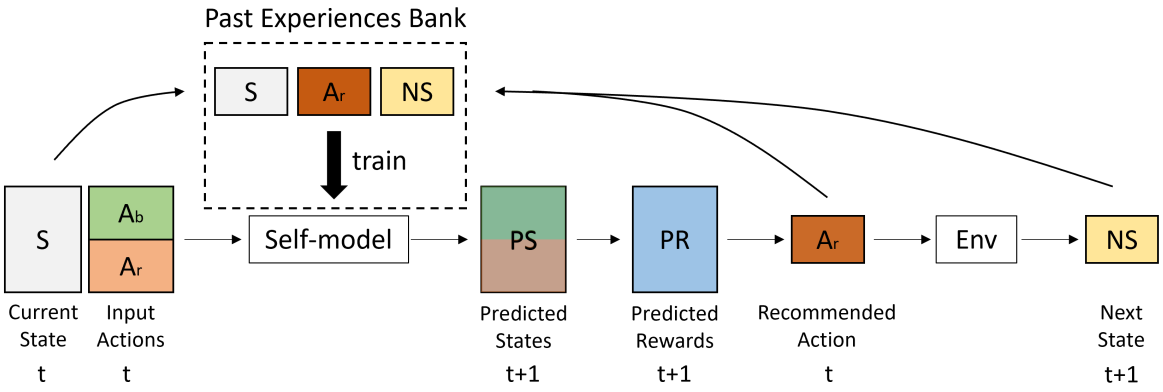


Figure 1: The pipeline begins with the robot current state being replicated to match the shape of the input action, which is then concatenated with the input action. Given that the batch size equals the number of actions, each state-action pair generates one Predicted State (PS) for the next time step t+1. Following this, the reward function computes the Predicted Reward (PR) using the PS. The action associated with the maximum PR is selected as the Recommended Action (Ar) and is subsequently executed in the real environment (Env) to yield the Ground Truth Next State (NS). Finally, the Quasi-static Self-Model is updated with the data consisting of the current State (S), Ar, and NS, setting the stage for improved state predictions in subsequent epochs.

ward. The recommended action is generated using a reward function that is dependent on the predicted state, reflecting the model's current understanding of the environment. Once determined, the recommended action is executed in a physical simulation environment (Env), which produces the ground truth next state.

In each training epoch, the algorithm synthesizes the current State, the recommended action, and the observed next state, utilizing this data to incrementally train and refine the self-model. The model starts with a reliance on Motor Babbling for its input actions due to the limited knowledge it possesses at the beginning.

As the algorithm progresses, it transitions to using a variation of the previous best actions, modified by adding Gaussian noise. This creates new actions that are similar to the previous best actions and allows the model to explore the space around known good actions. This method ensures consistency by limiting the range of exploration. Various traditional exploitation-

exploration tradeoff strategies can be used.

Through successive iterations and data collection, the quasi-static self-model continuously improves its predictive accuracy of the next state, enhancing its overall effectiveness and reliability over time. The data is collected by the MPC controller. In contrast to the uniform action space distribution produced by the initial motor babbling, the MPC provides a data distribution focused on robot locomotion.

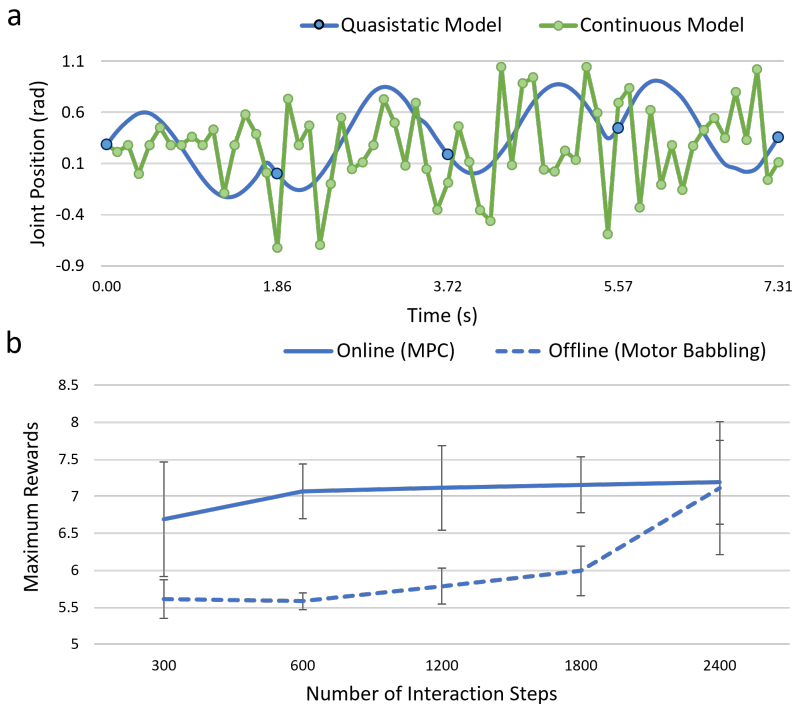


Figure 2: **a. Comparison of Joint Position Data for Quasi-static Self-Modeling vs. Continuous Model:** This figure illustrates the joint position (in radians) over time for the Quasi-static Self-Modeling (QSM) approach versus a continuous model during sinusoidal locomotion tasks. The x-axis represents time, while the y-axis represents the joint position in radians. Each circle represents a single sample point. **b. Performance comparison with respect to the number of utilized samples:** The blue curve signifies our method, reaching over 7.0 rewards after 600 steps and subsequently converging, while the offline method without MPC to update the self-model during data collection requires 2400 steps to achieve comparable performance using motor babbling. Both methods use the same quasi-static action. The models were trained 3 times and used to perform forward locomotion via model predictive control 1000 times, illustrating the superior efficiency of our method in terms of sample utilization.

In comparison to continuous models, our QSM approach demonstrates significant advantages in handling data for locomotion tasks. Our framework is structured as a Markov Decision Process (MDP) (*46*), but with a sequence of steps amalgamated into a dynamic process generated by a sinusoidal function. This adjustment allows for smoother and more natural robot movement, inspired by the principles of Central Pattern Generator (*47*).

We present a plot comparison illustrating the data utilized for both the quasi-static and continuous models during sinusoidal locomotion. Due to the randomization of parameters in the sinusoidal function, the QSM data is noticeably less fluctuant than that of the continuous model (see Fig.2 a). The parameter randomization reduces high-frequency oscillations and results in a smoother, more stable learning process. In contrast, the data for the continuous model (represented by the more oscillating curve) exhibits greater variability with high-frequency changes, denoting a more dynamic state and posing higher computational demands and potential instability.

Robot complexity and the utility of Quasi-static Self-Modeling

Researchers have dedicated immense effort toward the development of robust gait policies that optimize the locomotion of their robots. Through the application of reinforcement learning algorithms, robots are now capable of achieving dynamic, stable, and efficient locomotion across a variety of challenging terrains and tasks. However, much of this work has been primarily focused on single, specific robotic platforms. The controllers and policies developed are typically tailored to the intricacies of a particular robot. Hence, their generalizability to other platforms with varying degrees of complexity remains an open question.

Robotic complexity, often characterized by factors such as the number of degrees of freedom, variety in actuator types, and the intricacy of sensorimotor coordination, can significantly influence the design, training, and performance of reinforcement learning algorithms. This

study sets out to explore these understudied areas, looking into the relationship between robotic complexity and model-based reinforcement learning. In doing so, we hope to provide insights and methodologies that can guide future research efforts and enable the broader application of reinforcement learning to increasingly complex-legged robotic systems.

In this study, we developed an experiment involving 80 distinct robotic systems. These systems are characterized by a variety of configurations and complexities, which will allow us to gain a broader and more nuanced understanding of the relationship between robotic complexity and the value derived from model-based training. The novelty of our approach is also shown in the small amount of data required for training our self-model. The data efficiency of our method outperforms other studies. For instance, the training of a Minitaur robot required a staggering 45,000 control steps in previous work by Yang et al., while Bellegarda et al. required 'a few million time steps' for training a Unitree A1 robot (*14, 48*). In contrast, we utilize a compact dataset of merely 600 steps to train our quasi-static self-model, which is substantially smaller by several orders of magnitude compared to the aforementioned studies.

To justify this choice of data volume, we conducted experiments where we trained the models with different datasets of varying sizes, ranging from 300 to 600, 1200, 1800, and 2400 steps. The experimental process commenced with the training of the models. Subsequently, the robots employed these models to predict the subsequent state using Model Predictive Control (MPC) controllers. For each experimental trial, the coefficients of the objective function were randomized 100 times, with the maximum reward logged for each randomization. To ensure the consistency and robustness of our results, this data collection, model training, and testing process was replicated three times for each data volume, thereby providing a comprehensive evaluation of the system's performance across multiple trials.

The results of these experiments are summarized in Figure 2b. Our findings indicated that a dataset comprising 600 steps was adequately large to enable the robot to execute basic forward

locomotion effectively.

Importantly, larger datasets did not yield a proportionate improvement in performance, suggesting that additional data beyond the 600-step threshold did not significantly contribute to model training. This not only verified the efficiency of our Quasi-static Self-Modeling approach but also informed our decision to utilize a 600-step dataset for the main experiments. Through this approach, we ensured that our models were trained using an optimal data volume, thus guaranteeing efficiency while retaining performance.

Results

Training a locomotion policy in the synthetic environment

,

In this section, we present the real-world results of combing the QSM with model-free reinforcement learning-Proximal Policy Optimization (PPO) algorithm to train a locomotion policy (*49*). We conduct experiments on a 12-degree-of-freedom (DOF) legged robot, as shown in Figure 3. Our goal is to train a predictive model for the robot and evaluate the forward locomotion policy learned with this model using Quasi-static Self-Modeling.

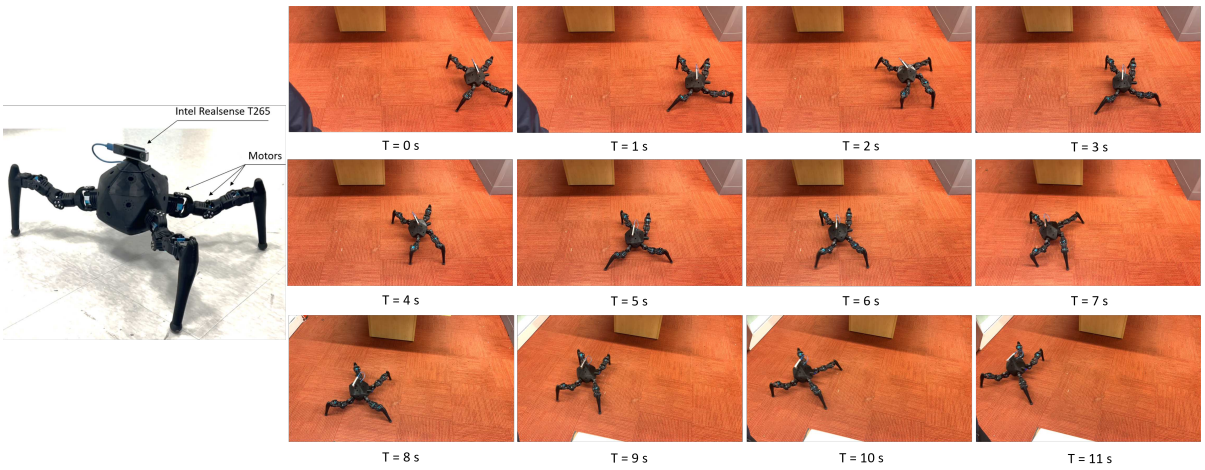


Figure 3: **Legged robot locomotion in the real world using PPO trained using the Quasi-static self-modeling.** Our legged robot platform has 12 motors using position control. Each leg has 3 DoF. The Intel Realsense tracking camera T265 provides the state of the robot. The right pictures show the robot doing forward locomotion in real-time on the carpet in our lab using the policy trained by model free RL (PPO) and quasi-static self-model.

Once the quasi-static self-model is established, it becomes a synthetic environment, predicting the next observation based on the state and action input. The trained model simulates the response of different actions from the robot, and the policy is trained within this synthetic environment using PPO.

Our method addresses the shortcomings of conventional reinforcement learning methods, presenting a data-efficient and effective solution to navigate complex dynamic environments. The robot locomotion video can be found in Movie 1. In real-world experiments, a robot trained by our method can reach a forward speed of 0.143m/s, while the baseline PPO cannot perform any noticeable locomotion at all.

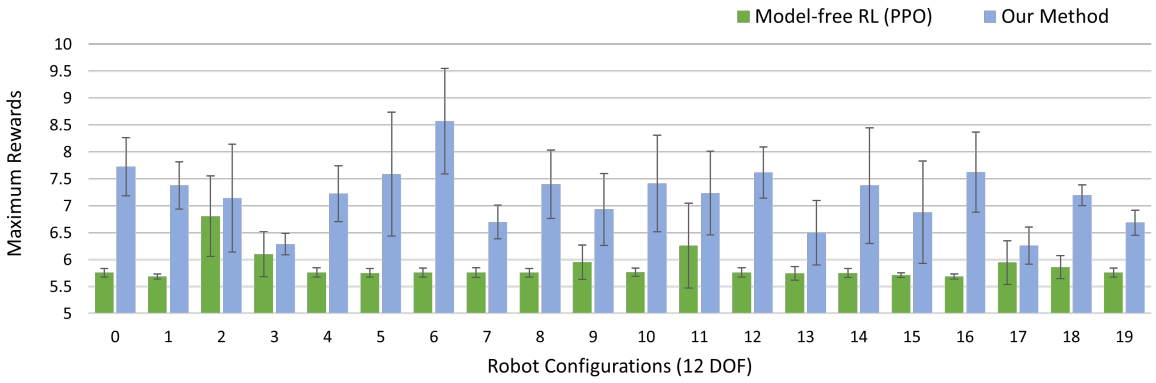


Figure 4: **Performance comparations of quasi-static Self-Modeling and PPO on Forward Locomotion.** This plot demonstrates the superior performance of our method (depicted in blue) against the Proximal Policy Optimization (PPO) baseline (shown in green) in various robot configurations, represented on the horizontal axis. The horizontal axis represents robot configurations, while the vertical axis is the maximum rewards obtained by the policy.

Quantitative evaluations for the scaling study were conducted in a simulation only (Figure 4). We used Gaussian noise to initialize the action space, generating 20 different morphological

configurations for the same joint hardware (see Movie 2). Each experiment used only 600 interaction steps in the simulated environment, as described in the previous section. Subsequently, we utilized the quasi-static self-model as a synthetic environment to facilitate the training of a forward locomotion policy via the Proximal Policy Optimization (PPO) algorithm. The transition to the training phase required no additional data.

In each experiment, we train the PPO with a quasi-static self-model for 24,000 interaction steps. We provide visual comparisons in the simulation in Movie 2 and Movie 3. The reward obtained by the robot is proportional to the forward amount and penalizes lateral offset and heading angle divergence. Through these experiments involving varied robot configurations and limited interaction steps, our QSM-trained robot consistently outperformed the baseline model-free RL (PPO). These performance differences were significant and consistent across all experiments, demonstrating the robustness and generalizability of our method across morphologies.

Accuracy Comparison: Quasi-static Self-Model vs. Continuous Models

In this experiment, we aimed to evaluate the performance of the quasi-static self-model in comparison to the continuous model in model-based reinforcement learning. Specifically, we investigated the advantage of the quasi-static self-model in achieving better state prediction accuracy using a 12-degree-of-freedom (DOF) legged robot.

To create a controlled and replicable environment for our investigations, we opted to conduct our experiments within a simulated platform. The initial phase of the experimental process involved the utilization of the motor babbling technique to compile a dataset comprising 600 pairs of observed states, actions, and ground truth of the next states for both the world model and the QSM. The world model (WM) requires the prediction of the full dynamic state, whereas the QSM targets the prediction of the relatively static state that emerges after the sinusoidal actions.

Table 1: Comparison of Prediction Errors between the Continuous Model (CM) and Quasi-static Self-Model (QSM)

		Δx(m)	Δy(m)	Δz(m)	Δroll(rad)	Δpitch(rad)	Δyaw(rad)
CM	Mean	4.35E-03	5.69E-03	1.21E-03	2.89E-01	1.54E-02	4.68E-02
	Std.	9.37E-04	2.73E-03	2.02E-04	3.66E-01	4.01E-03	2.62E-03
	Min.	3.56E-03	3.10E-03	1.01E-03	2.62E-02	1.06E-02	4.45E-02
	Max.	5.66E-03	9.46E-03	1.49E-03	8.06E-01	2.04E-02	5.05E-02
QSM	Mean	2.91E-03	3.64E-03	9.47E-05	2.96E-03	2.83E-04	4.00E-02
	Std.	4.00E-04	6.38E-04	9.83E-06	7.21E-04	3.82E-05	3.27E-03
	Min.	2.56E-03	2.82E-03	8.77E-05	2.31E-03	2.52E-04	3.76E-02
	Max.	3.47E-03	4.38E-03	1.09E-04	3.97E-03	3.37E-04	4.47E-02

To compare the efficacy of the two models, we adopted the model predictive control (MPC) methodology, utilizing the trained models to execute a forward locomotion task. This task necessitated the prediction of subsequent states by inputting the current state and potential actions into both the WM and QSM. The choice of the subsequent action by the robot was dictated by the rewards calculated based on the predicted states. To ensure the reliability of our results, the experiment was executed 100 times. Each run involved the collection of prediction errors and the ground truth. Table 1 presents the prediction errors for both models regarding the six robot states including Δx, Δy, Δz, Δroll, Δpitch, and Δyaw. For each of these parameters, we assessed the mean, standard deviation, minimum, and maximum errors in order to provide a comprehensive overview of the model performance.

The results clearly demonstrate the advantage of our method in achieving better state prediction accuracy across all state variables. The experimental results not only demonstrate the advantage of quasi-static self-modeling in state prediction but also highlight through incorporating the self-model into model-based reinforcement learning, the robot can achieve more accurate predictions, leading to more efficient robot locomotion policy training.

Quasi-static Self-Model Utility Scales with Complexity

To quantify the relationship between robot complexity and the usefulness of quasi-static self-models, we compare the performance of the model-free RL (PPO) to our method. The experiments were conducted on a set of 80 different robots, each having an overall morphology identical to the 12 degrees-of-freedom (DOF) legged robot discussed earlier. To vary the DOF and control the variable of the robot systems, we selectively froze motor joints. For instance, for a 4 DOF robot system, only four active motors were allowed, with the remaining motors kept inactive. For the 12 DOF robot, we manipulate the action space to produce different configurations. To ensure that these configurations are feasible for locomotion tasks, we maintain symmetry in the robot actuator selection. For each robot, we compared the performance of a policy trained via our method with that trained via the baseline PPO using the 600 interactive steps. The PPO policy trained by our method is trained for 24k steps based on the synthetic environment provided by the quasi-static self-model without additional data.

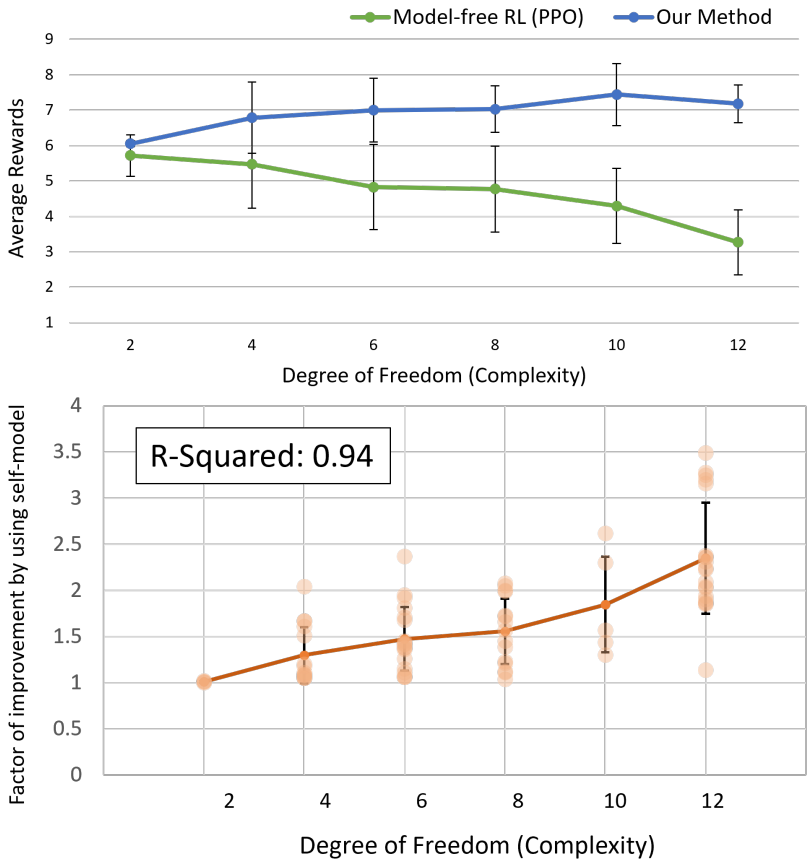


Figure 5: **The results of comparisons:** a) Comparative performance of our method (blue curve) versus the baseline (green curve) with respect to varying degrees of freedom. Our method consistently delivers increasing average rewards, signifying stable learning effectiveness, while the baseline PPO declines performance as system complexity increases. b) Scatter plot with overlaid curve demonstrating the improvement factor achieved by using our self-model method compared to the baseline PPO method across multiple robots. Each scatter point represents the ratio of rewards achieved by our method to the baseline, calculated over three runs for each robot. The distribution includes five robots for 2 and 10 DOF, fifteen robots for 4 and 8 DOF, and twenty robots for 6 and 12 DOF.

As depicted in Figure 5a, our method can provide stable training even as the robot complexity increases. On the contrary, the PPO model-free algorithm struggles with limited data, resulting in an insufficiently trained policy. Consequently, when the complexity of the robot increases, and the action space expands, the average rewards for the baseline PPO method tend

to decline. Our quasi-static self-model method maintains the performance, demonstrating its robustness and data efficiency in handling more complex robotic systems with limited data. In Figure5b. We calculate the factor of improvement by using quasi-static self-models for 80 robots. All the data points are calculated three times from data collection to the test experiments using MPC. The ascending curve exhibits a strong positive correlation with an R-squared value of 0.94, indicating that our quasi-static self-modeling method becomes increasingly beneficial for more complex robotic systems. Further visual evidence of our method's efficacy is provided in Movies 4 and 5, which showcase the locomotion results for all the robots involved in this experiment.

Discussion

We demonstrated the effectiveness of our quasi-static self-modeling framework through experiments on a 12-degree-of-freedom legged robot. The results showed that our method outperformed the baseline model-free Proximal Policy Optimization (PPO) algorithm in terms of forward locomotion. The quasi-static self-modeling approach allowed the robot to achieve a forward speed of 0.143 m/s, while the baseline PPO failed to produce any locomotion.

We also evaluated the performance of the quasi-static self-model in predicting robot states and found that it outperformed a traditional world model in terms of state prediction accuracy.

Furthermore, we investigated the relationship between robotic complexity and the utility of quasi-static self-modeling. Our experiments with 80 different robot configurations demonstrated that the utility of the quasi-static self-model increased with the complexity of the robot. This finding highlights the potential of our method for training complex robotic systems with limited data.

However, there are several limitations and potential areas for future improvement. First, our experiments focused on forward locomotion, and it would be interesting to investigate the appli-

cability of our approach to other locomotion tasks or complex robotic behaviors. Additionally, while our method demonstrated robust performance in different robot configurations, further research is needed to explore its scalability to more advanced-legged robot platforms with torque control. Moreover, the generalization capability of policies trained using our method to unseen environments or variations in robot morphology should be examined.

In conclusion, our quasi-static self-modeling framework presents a novel approach that bridges the gap between self-modeling and reinforcement learning, enabling efficient and stable policy learning for complex robotic systems. The quasi-static self-modeling approach offers a promising solution for data-efficient and stable training of robots in real-world environments. By leveraging self-modeling, robotic systems can acquire a deeper understanding of their own dynamics and kinematics, leading to more intelligent and adaptive behavior in real-world environments.

Supplementary materials

Legged Robot Configurations

In this supplementary section, we provide detailed information about the 80 different configurations of the robot used in our experiments. These configurations were designed to vary the degrees of freedom (DOF) and control variables of the robot systems, allowing us to explore the relationship between robotic complexity and the utility of quasi-static self-modeling.

Table 2: Robot name and activated motor identification

Name	Motor ID	Name	Motor ID	Name	Motor ID
200	[0, 9]	600	[0, 1, 2, 9, 10, 11]	800	[0, 2, 3, 5, 6, 8, 9, 11]
201	[4, 7]	601	[0, 1, 3, 6, 9, 10]	801	[0, 1, 2, 3, 6, 9, 10, 11]
202	[1, 10]	602	[2, 4, 5, 7, 8, 11]	802	[0, 1, 2, 4, 7, 9, 10, 11]
203	[5, 8]	603	[0, 3, 4, 6, 7, 9]	803	[0, 1, 2, 5, 8, 9, 10, 11]
204	[2, 11]	604	[0, 1, 5, 8, 9, 10]	804	[0, 3, 4, 5, 6, 7, 8, 9]
400	[0, 3, 6, 9]	605	[0, 2, 3, 6, 9, 11]	805	[1, 3, 4, 5, 6, 7, 8, 10]
401	[0, 4, 7, 9]	606	[0, 2, 4, 7, 9, 11]	806	[2, 3, 4, 5, 6, 7, 8, 11]
402	[0, 5, 8, 9]	607	[0, 2, 5, 8, 9, 11]	807	[0, 1, 3, 4, 6, 7, 9, 10]
403	[1, 3, 6, 10]	608	[1, 2, 3, 6, 10, 11]	808	[0, 1, 3, 5, 6, 8, 9, 10]

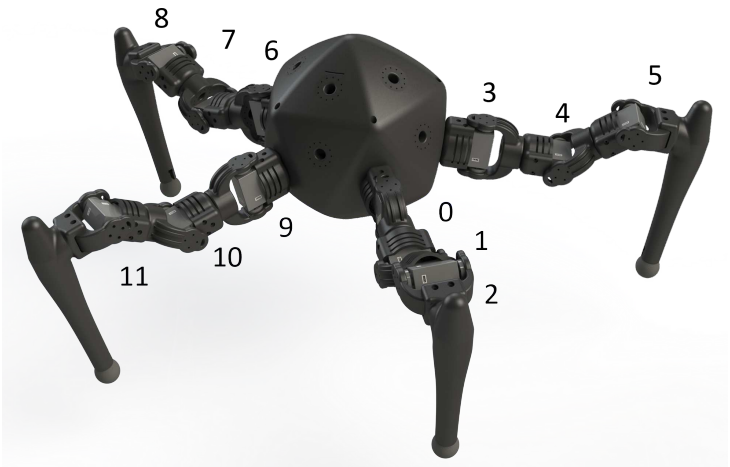


Figure 6: **Robot motor identification.** Each leg has three motors.

For each configuration, we describe the specific motor joints and their activation status. By selectively freezing motor joints and maintaining symmetry in the selection of active motors, we ensured that the configurations were feasible for locomotion tasks while varying the complexity of the robot. Fig. 6 shows the motor identification of our legged robot. Table 2 lists the activated motor identification for the 60 robots which has frozen motors. The remaining 20 robots use all 12 motors and Gaussian noise 0.2 for the action space to create new configurations.

Architecture and usage of the model

The neural network model used in our quasi-static self-modeling framework is implemented as a neural network with five fully connected layers. The input layer takes the concatenated state and action as input and is followed by three hidden layers with a size of 128 each. The final output layer produces the predicted next state vector. The activation function used in the hidden layers is the hyperbolic tangent. During the forward pass, the input is passed through the layers, applying the activation function after each hidden layer. The output of the final layer represents the predicted next state. The loss function is calculated as the mean squared error between the predicted next state and the target next state. This architecture enables the quasi-static self-model to learn the relationships between the current state, action, and subsequent state changes.

Data Representation and Reward Function

In this section, we discuss the data representation used in our quasi-static self-modeling framework for robot locomotion.

The state representation refers to how we encode the information about the robot's current state. In our framework, the state representation consists of an 18-size vector:

- $\Delta x, \Delta y, \Delta z$: These variables represent the changes in the robot's position along the x, y, and z axes, respectively.
- $\Delta roll, \Delta pitch, \Delta yaw$: These variables capture the changes in the robot's orientation or roll, pitch, and yaw angles.
- Joint angles: The vector includes the joint angles of the robot, providing information about the motor position.

The action representation refers to how we encode the robot's actions in a format suitable for learning. In our quasi-static self-modeling framework, the action representation consists of 16 parameters that control the generation of a sinusoidal curve. This action representation enables the robot to produce smooth trajectories during locomotion.

The reward function for the MPC controller and PPO policy training is defined as:

$$r = a \cdot \Delta y - b \cdot |\Delta \text{yaw}| - c \cdot |\Delta x| \tag{1}$$

Here Δy is the change in the forward axis, Δyaw is the change in yaw (rotation around the vertical axis) and Δx is the change in the lateral axis. a, b and c are constants that control the relative importance of moving forward, minimizing yaw change, and minimizing lateral movement. During PPO policy training, a, b and c are 3, 1, 0.5.

Parallelization Strategy for Efficient Training and Enhanced Stability

Our quasi-static Self-Modeling approach leverages the synthetic environment not only for efficient training but also for facilitating easy parallelization of the training procedures. This parallelization capability is particularly advantageous in mitigating the instability issues commonly associated with Proximal Policy Optimization (PPO).

In traditional reinforcement learning, PPO sometimes fails to produce an effective policy, necessitating complete retraining and resulting in substantial real-world data waste. Our method, however, allows for the simultaneous training of multiple PPO instances, each under a different initial seed within the synthesized environment. This parallelization strategy accelerates the training process and results in the generation of multiple unique policies without requiring additional data.

The value of this strategy extends beyond simply speeding up the training process; it also presents a crucial advantage in handling the inherent instability often encountered with model-free reinforcement learning. Policies exhibiting early signs of failure can be promptly identified

and discontinued, preserving vital computational resources. Concurrently, more promising policies can be selected and pursued for further training, ensuring the efficient use of resources and optimizing the chances of achieving effective locomotion policies.

Movies and Code

Movie 1: A comparison between our method and the Proximal Policy Optimization (PPO) baseline in real-world environment.

Movie 2 and Movie 3: Demonstrations of 20 different 12-degree-of-freedom (DOF) robots using our method and the PPO baseline in a simulation environment. Each robot experiment was repeated three times.

Movie 4 and Movie 5: Simulation Results for 2 DOF to 12 DOF Robots These videos demonstrate the simulation results obtained from 60 robots, ranging from 2 DOF to 12 DOF, using our method and the PPO baseline. Each robot experiment was conducted three times.

The source code for our method and experiments is available on GitHub at: https://github.com/H-Y-H-Y-H/OSM

Acknowledgements

This work was supported in part by the US National Science Foundation (NSF) AI Institute for Dynamical Systems (DynamicsAI.org), grant 2112085

Competing Interests

The Authors declare no Competing Financial or Non-Financial Interests.

Data Availability

All data generated during this study are available, including the training data, trained models, and evaluation data. (*50*).

Author Contributions

Y.H., R.K., and H.L. proposed the research and developed the main idea. Y.H. presents algorithm designs, implementations, simulations, and real-world experiments. H.L., B.C., and P.W. provided deep insights and guidance on the algorithm and experiment design. Y.H. and H.L. performed numerical analysis. Y.H. and H.L. wrote the main manuscript text, and Y.H. prepared all the figures and videos. All authors reviewed the manuscript and provided feedback.

References

1. G. G. Gallup Jr, "Self-awareness and the emergence of mind in primates," *American Journal of Primatology*, vol. 2, no. 3, pp. 237–248, 1982.

2. P. Rochat, "Five levels of self-awareness as they unfold early in life," *Consciousness and cognition*, vol. 12, no. 4, pp. 717–731, 2003.

3. B. Chen, R. Kwiatkowski, C. Vondrick, and H. Lipson, "Fully body visual self-modeling of robot morphologies," *Science Robotics*, vol. 7, no. 68, p. eabn1944, 2022.

4. A. Dearden and Y. Demiris, "Learning forward models for robots," in *IJCAI*, vol. 5, p. 1440, 2005.

5. D. M. Wolpert, R. C. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends in cognitive sciences*, vol. 2, no. 9, pp. 338–347, 1998.

6. J. C. Bongard, "Accelerating self-modeling in cooperative robot teams," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 321–332, 2008.

7. R. Kwiatkowski and H. Lipson, "Task-agnostic self-modeling machines," *Science Robotics*, vol. 4, no. 26, p. eaau9354, 2019.

8. Y. Hu, B. Chen, and H. Lipson, "Self-supervised robot self-modeling using a single egocentric camera," 2023.

9. R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

10. L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

11. J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

12. Á. Belmonte-Baeza, J. Lee, G. Valsecchi, and M. Hutter, "Meta reinforcement learning for optimal design of legged robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12134–12141, 2022.

13. Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath, "Reinforcement learning for robust parameterized locomotion control of bipedal robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2811–2817, IEEE, 2021.

14. Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data efficient reinforcement learning for legged robots," in *Conference on Robot Learning*, pp. 1–10, PMLR, 2020.

15. T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine, "Learning to walk via deep reinforcement learning," *arXiv preprint arXiv:1812.11103*, 2018.

16. N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3, pp. 2619–2624, IEEE, 2004.

17. R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3, pp. 2849–2854, IEEE, 2004.

18. J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

19. Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, "Feedback control for cassie with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1241–1246, IEEE, 2018.

20. X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," *arXiv preprint arXiv:2004.00784*, 2020.

21. B. Katz, J. Di Carlo, and S. Kim, "Mini cheetah: A platform for pushing the limits of dynamic quadruped control," in *2019 international conference on robotics and automation (ICRA)*, pp. 6295–6301, IEEE, 2019.

22. V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

23. N. Rudin, H. Kolvenbach, V. Tsounis, and M. Hutter, "Cat-like jumping and landing of legged robots in low gravity using deep reinforcement learning," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 317–328, 2021.

24. E. Vollenweider, M. Bjelonic, V. Klemm, N. Rudin, J. Lee, and M. Hutter, "Advanced skills through multiple adversarial motion priors in reinforcement learning," *arXiv preprint arXiv:2203.14912*, 2022.

25. J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," *arXiv preprint arXiv:1901.07517*, 2019.

26. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

27. T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

28. R. Nian, J. Liu, and B. Huang, "A review on reinforcement learning: Introduction and applications in industrial process control," *Computers & Chemical Engineering*, vol. 139, p. 106886, 2020.

29. N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, pp. 91–100, PMLR, 2022.

30. G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

31. P. Kormushev, S. Calinon, and D. G. Caldwell, "Reinforcement learning in robotics: Applications and real-world challenges," *Robotics*, vol. 2, no. 3, pp. 122–148, 2013.

32. P. Ladosz, L. Weng, M. Kim, and H. Oh, "Exploration in deep reinforcement learning: A survey," *Information Fusion*, 2022.

33. T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker, *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

34. L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, *et al.*, "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.

35. T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, "Benchmarking model-based reinforcement learning," *arXiv preprint arXiv:1907.02057*, 2019.

36. M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in neural information processing systems*, vol. 32, 2019.

37. A. Plaat, W. Kosters, and M. Preuss, "High-accuracy model-based reinforcement learning, a survey," *Artificial Intelligence Review*, pp. 1–33, 2023.

38. A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine, "Visual reinforcement learning with imagined goals," *Advances in neural information processing systems*, vol. 31, 2018.

39. K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

40. T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

41. J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee, "Sample-efficient reinforcement learning with stochastic ensemble value expansion," *Advances in neural information processing systems*, vol. 31, 2018.

42. S. Park and S. Levine, "Predictable mdp abstraction for unsupervised model-based rl," *arXiv preprint arXiv:2302.03921*, 2023.

43. T. Yu, A. Kumar, R. Rafailov, A. Rajeswaran, S. Levine, and C. Finn, "Combo: Conservative offline model-based policy optimization," *Advances in neural information processing systems*, vol. 34, pp. 28954–28967, 2021.

44. T. Yu, G. Thomas, L. Yu, S. Ermon, J. Y. Zou, S. Levine, C. Finn, and T. Ma, "Mopo: Model-based offline policy optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14129–14142, 2020.

45. P. Abbeel, M. Quigley, and A. Y. Ng, "Using inaccurate models in reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, pp. 1–8, 2006.

46. R. Bellman, "A markovian decision process," *Journal of mathematics and mechanics*, pp. 679–684, 1957.

47. A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

48. G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust high-speed running for quadruped robots via deep reinforcement learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10364–10370, IEEE, 2022.

49. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

50. Y. Hu, R. Kwiatkowski, B. Chen, P. M. Wyder, and H. Lipson, "Dataset for the paper titled Scalable Quasi-static Self-modeling for Physical Legged Locomotion," 6 2023.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- video.zip