Learning Spring Mass Locomotion: Guiding Policies With a Reduced-Order Model

Kevin Green, *Graduate Student Member, IEEE*, Yesh Godse, Jeremy Dao, Ross L. Hatton, *Member, IEEE*, Alan Fern, and Jonathan Hurst, *Senior Member, IEEE*

***Abstract*—In this letter, we describe an approach to achieve dynamic legged locomotion on physical robots which combines existing methods for control with reinforcement learning. Specifically, our goal is a control hierarchy in which highest-level behaviors are planned through reduced-order models, which describe the fundamental physics of legged locomotion, and lower level controllers utilize a learned policy that can bridge the gap between the idealized, simple model and the complex, full order robot. The high-level planner can use a model of the environment and be task specific, while the low-level learned controller can execute a wide range of motions so that it applies to many different tasks. In this letter, we describe this learned dynamic walking controller and show that a range of walking motions from reduced-order models can be used as the command and primary training signal for learned policies. The resulting policies do not attempt to naively track the motion (as a traditional trajectory tracking controller would) but instead balance immediate motion tracking with long term stability. The resulting controller is demonstrated on a human scale, unconstrained, untethered bipedal robot at speeds up to 1.2 m/s. This letter builds the foundation of a generic, dynamic learned walking controller that can be applied to many different tasks.**

***Index Terms*—Humanoid and bipedal locomotion, legged robots, reinforcement learning.**

I. Introduction

A POWERFUL approach to control agile and dynamic legged robots is to use a control hierarchy that combines specific domain knowledge of legged locomotion with the power of deep reinforcement learning. The long term goal is to enable robots to be able to navigate quickly through previously unseen environments with agility that approaches or exceeds that of humans and animals. The control hierarchy should consist of a low-level walking controller generated through reinforcement learning that can account for and exploit the passive dynamics of the physical robot. This low-level controller receives motion commands from a terrain-aware motion planner. The commands from the planner must be rich enough to sufficiently direct the walking controller while still being as simple as possible. This letter focuses on the learned *controller* and its interface, so we elect to use a library of precomputed motions (Fig. 1).

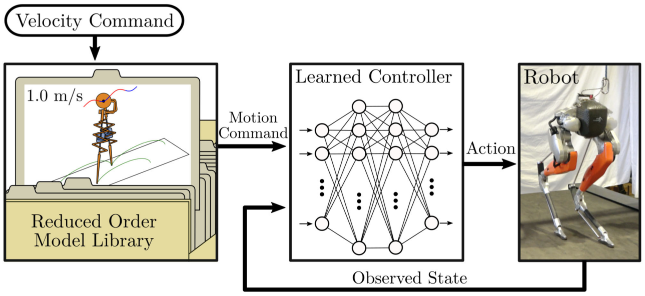

Fig. 1. Our proposed control hierarchy which demonstrates a learned controller for a legged robot that is commanded using reduced-order model motions. In future work, this library of reduced order model motions can be replaced by a dynamic motion planner.

Learned controllers have incredible potential to create dynamic locomotion, but to be integrated into a control hierarchy we need an effective control interface. Dynamic legged locomotion is by its nature underactuated, hybrid, unstable, nonlinear, and must be able to operate with significant ground uncertainty. These challenges may be addressed by deep neural networks acting as controllers because of their ability to encode highly nonlinear control policies. However, if we would like to develop more complex behaviors such as autonomous navigation through unknown, obstacle filled environments, we will need to extend this approach. It may be possible to expand the learning problem so that the same policy that dynamically controls the robot also interprets the world around it and chooses how to move to the goal, but we choose not to take this approach because of the challenge of generalization to new tasks, sensors and environments. Instead, we seek to create a learned controller that can be directed by other intelligent systems in a modular hierarchy. Recent work explores the use of hierarchical learned control structures to quadrupedal locomotion. Some methods use both a high-level learned policy and a low-level learned policy [1], [2]. Another method combines a low-level model based controller with a high-level, learned gait planner [3].

Reduced-order models of locomotion capture the core dynamics of locomotion which make them a compelling control interface. The most common models used to plan motion for

Manuscript received October 14, 2020; accepted February 18, 2021. Date of publication March 17, 2021; date of current version April 5, 2021. This letter was recommended for publication by Associate Editor L. Righetti and Editor A. Kheddar upon evaluation of the reviewers' comments. This work was supported in part by DARPA Contract W911NF-16-1-0002, NSF Grant No. CMMI-1653220, and NSF Grant DGE-1314109. *(Corresponding author: Kevin Green.)*

The authors are with the Collaborative Robotics, and Intelligent Systems Institute, Oregon State University, Corvallis, Oregon 97331 USA (e-mail: greenkev@oregonstate.edu; godsey@oregonstate.edu; jeremydao913@gmail.com; ross.hatton@oregonstate.edu; alan.fern@oregonstate.edu; jonathan.hurst@oregonstate.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2021.3066833, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3066833

bipedal robots are inverted pendulum models, which consist of a point mass and massless legs that can apply forces through ground contact [4]–[6]. These models describe the underactuation of dynamic walking as well as the discrete choice of foot step locations while remaining simple enough to plan with in real time. The spring loaded inverted pendulum (SLIP) model is particularly applicable for agile locomotion because with simple, feed-forward policies it demonstrates strong stabilizing effects [7]–[9]. Many agile robots closely resemble the SLIP model [10], [11] or are designed with SLIP locomotion as a goal [12], [13], which motivates our choice to use an actuated variation of the SLIP model in this letter.

In this letter we present a method of using reduced-order models of walking to direct high quality, transferable walking controllers and demonstrate its effectiveness on a Cassie series robot from Agility Robotics. We use the bipedal actuated SLIP model as the reduced-order model of walking to create a library of gaits across different speeds. The gaits are optimized using a direct collocation method to create energetically optimal walking cycles. Following these trajectories makes up 70% of the reinforcement learning reward while 20% is for foot orientation and 10% is for smooth actions. The resulting controller produces visually natural motions with clear correspondence to the reference trajectories. These results show that reduced-order model trajectories are useful tools in training and controlling learned walking controllers.

II. Background

Reinforcement Learning is a learning framework in which agents learn what actions to take in order to maximize their cumulative future reward. Policy gradient methods, such as Proximal Policy Optimization (PPO) [14], are a popular choice of reinforcement learning algorithms that have been successfully applied to generate control policies for robotic systems, including legged robots [15], [16]. An important part of using reinforcement learning to solve complex control problems is the design of the reward function evaluated after each agent-environment interaction. Most applications of reinforcement learning to legged locomotion employ heuristic reward functions to produce walking behavior [16], [17]. Though this method has been effective, it is often hard to describe a desired motion through objective, generic reward functions. The definition must be sufficiently detailed to prevent maladaptive policies from learning motions that exploit features of the reward or simulator and do not accomplish the underlying goal when transferred to hardware. Researchers often use long and complex reward functions to prevent these maladaptive policies; [17] uses 8 different reward terms for a single walking task.

A different approach is to use a single expert trajectory as a reference motion [15]. The reward function encourages motions that are close to the specified expert trajectory, which can result in a policy that closely recreates the desired motion. The use of the reference information discourages exploitative policies since the desired motion is now densely and properly represented. When a reference motion reward makes up a significant portion of the total reward, it severely restricts the space of solutions to be those near the reference motion. For some tasks this restriction is unacceptable; however, for bipedal locomotion restricting the final behavior to resemble a normal walking gait is actually preferable because it disincentives strange, exploitative behaviors.

We note that the method from [15] used a single reference trajectory. To provide effective information for a variety of speeds, this single trajectory was "stretched" and "compressed" to higher and lower speeds, sometimes creating trajectories that were physically infeasible. These infeasible trajectories may harm the learning problem by making the trajectory matching reward signal conflict with the dynamics of the system. Our work mitigates this conflict by using reduced order model trajectories with inverse kinematics to produce feasible walking trajectories for every training speed. Furthermore, the use of a reduced-order model allows for greater flexibility in the control system, by allowing future work to use the reduced-order model trajectories as a form of higher level planning.

III. The Control Hierarchy

We created a control hierarchy (Fig. 2) to allow us to train and test a walking controller that utilized reduced-order model motion. The only external input to the system is a human operator's forward velocity command. Internally, a periodic clock increments forward through the walking cycle. The velocity command and clock are inputs to a library of reduced-order model motions (§III-A). The library returns the positions and velocities of the reduced-order model's body and feet for use as input to the learned policy. Additionally, the library contains a set of robot-specific motor angles that correspond to a robot pose that match the body and foot positions. The learned policy is evaluated and the output is summed together with the reference joint angles to form the motor proportional-derivative (PD) command. These commanded angles are sent to the high frequency control loop (§III-B). This control loop evaluates the PD controller, sends torques to the motors, measures robot sensors, filters sensor data, and estimates the full state of the robot. This structure is utilized not just in hardware but also in the simulation environment we use to train the learned policy (π).

A. The Reduced-Order Model Library

Our motion library consists of task space (body and foot) trajectories for periodic walking over a range of forward speeds. Body and foot trajectories will be used as an input to the learned policy as well as the majority of the reward signal. These trajectories will not be strictly dynamically feasible on the full order robot, but they should describe a nearly feasible center of mass motion that can be produced by valid ground reaction forces at the feet. To create each trajectory we optimize the reduced-order model and augment it with a minimum-jerk swing leg profile. We additionally calculate motor angles for each motion through offline inverse kinematics to use as a feed forward term into the PD controller.

The motions in the library are energetically optimal periodic gaits of the 3D, bipedal actuated SLIP model. This model

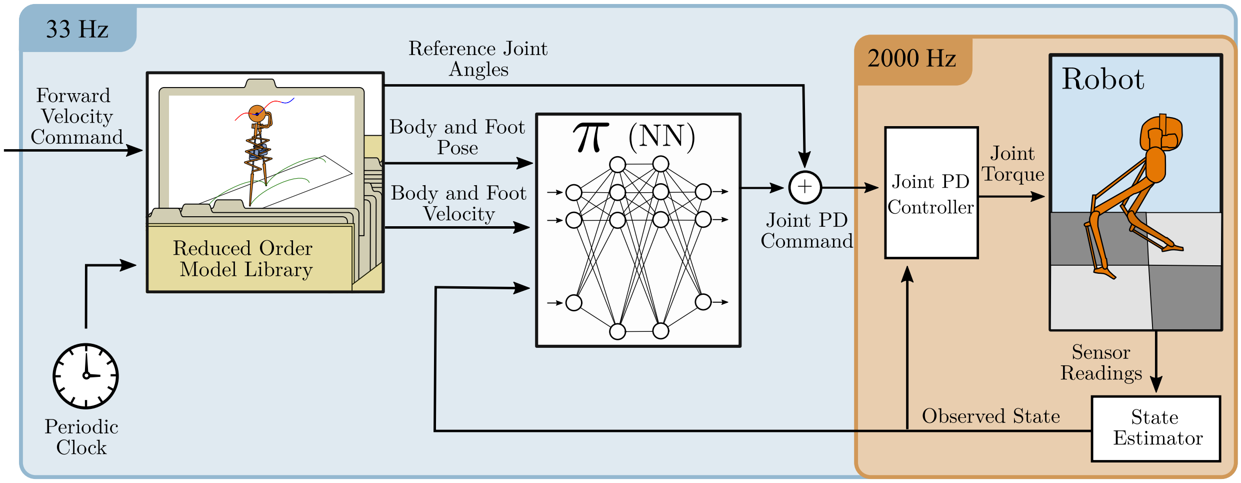

Fig. 2. The control diagram for both learning in simulation and running on hardware. At a relatively slow 33 Hz, the library is sampled and the learned policy is evaluated. At a much faster 2000 Hz, the joint PD controller is evaluated, the commanded torques are sent to the motors, and the state estimator is updated. The inputs to the learned policy are only the reduced-order model motion and the state estimate.

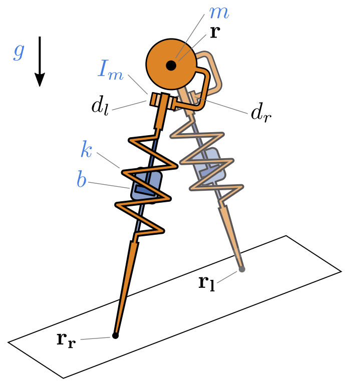

Fig. 3. The 3D, bipedal actuated spring loaded inverted pendulum model which we optimize to create walking gaits for our motion library. This model has a point mass body with no rotational inertia and two massless, compliant, actuated legs.

TABLE I
STATES, PARAMETERS AND CONTROL INPUTS FOR THE ACTUATED SLIP MODEL USED TO GENERATE THE LIBRARY OF MOTIONS

	Symbol	Value and Unit	Description
States	$\mathbf{r}$	- $[m,m,m]$	3D body position
	d_l	- $[m]$	Left leg actuator setpoint
	d_r	- $[m]$	Right leg actuator setpoint
Parameters	m	30 $[kg]$	Body mass
	k	3000 $[N/m]$	Leg spring stiffness
	b	2 $[Ns/m]$	Leg spring damping
	I_m	10 $[kg]$	Leg actuator linear inertia
	g	9.81 $[m/s]$	Gravitational acceleration
Continuous Inputs	u_l	- $[m/s^2]$	Left actuator acceleration
	u_r	- $[m/s^2]$	Right actuator acceleration
Discrete Inputs	$\mathbf{r_l}$	- $[m,m,m]$	Left foot stance position
	$\mathbf{r_r}$	- $[m,m,m]$	Right foot stance position

consists of a point mass body and two massless legs (Fig. 3). Parameters of the model were chosen to closely resemble the Cassie robot, see Table I. Each leg has a extensible actuator in series with a spring and a damper. The trajectory optimization method we use is a direct collocation method where the state and inputs are discretized and dynamics are enforced through equality constraints between sequential states [18]. Our implementation is not contact-invariant so we needed to specify the contact sequence. We define a walking contact sequence made up of alternating single stance and double stance phases. We vary the average forward velocities from 0 to 2 m/s in steps of 0.1 m/s as an equality constraint on the final state. The energetic objective function we use is

$$f = \int_0^T (u_l^2 + u_r^2)dt \tag{1}$$

where u_l and u_r are the accelerations of the leg actuators. This cost function represents the resistive losses in an electric motor when it applies forces to accelerate the inertia of its rotors. We ensure kinematic feasibility by including a conservative constraint on the maximum and minimum leg extension. To generate the swing foot motions we calculated minimum integrated jerk squared motion profiles. These profiles connect the footholds from the trajectory optimization with a specified vertical clearance height at the midpoint. We use a modified version of COALESCE [19] to generate the problem and its analytical gradients and IPOPT [20] to solve the problem.

An example optimal motion with a mean velocity of 1.0 m/s is shown in Fig. 4. The different phases can be seen in the main body's path where it reaches its peak height in single stance and the minimum height in double stance. The feet follow smooth paths with the 0.2 m specified vertical clearance height.

B. High Frequency Control Loop

The task space motions of the reduced-order model are inputted to a learned policy running at 33 Hz which outputs delta motor positions δ_a. These deltas are added to the baseline joint angles from the library, $\hat{a}$, to create the final command for the PD controller, $a = \hat{a} + \delta_a$. The use of joint angle baseline actions was used in prior work on learned walking controllers for Cassie [21]. The actions are passed to a joint level PD controller

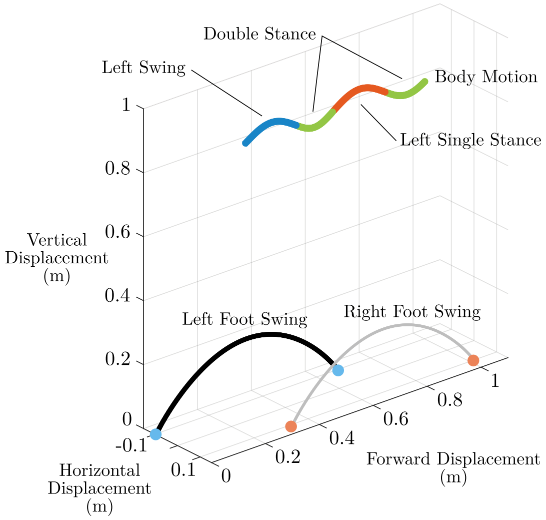

Fig. 4. The optimized task space motion of the reduced-order model for 1.0 m/s walking. The hybrid phase is shown through the color of the body trajectory. This motion together with its velocities make up the motion library.

with fixed feedback gains running at 2 kHz. Note that the learned policy only outputs position targets while the velocity targets are always set to zero. This means the proportional-derivative controller should more accurately be called a proportional-damping controller. We choose this structure as previous work has shown learning PD targets to be easier and produces higher quality motions [22].

IV. Reinforcement Learning

The optimal learned controller should be able to capture the most important features of the reduced-order model's motion and translate them into a stable walking behavior that functions on hardware.

A. Problem Formulation

The inputs to the policy are the estimated robot state and the desired positions and velocities of the robot's pelvis and feet. The estimated robot state contains the pelvis height, orientation, translational velocity, rotational velocity, and translational acceleration in addition to the positions and velocities of the actuated and unactuated joints on Cassie. This state estimate together with the commanded pose and velocity from the reduced-order model library form a $64D$ input space. The output of the learned policy is a $10D$ vector containing motor position targets for each of Cassie's actuated joints.

We learn control policies by using a simulated model of Cassie[1] in the MuJoCo Physics simulator [23]. Our dynamic model of Cassie includes the reflected inertia of each motor (defined as "armature" in MuJoCo). We also attempt to model actuator delay by limiting when new torque commands are actually executed. Desired torques only take effect 0.003 seconds (6 time steps of the high frequency, PD control loop) after being "sent" to the simulator. We believe these extra facets of the model help improve the policy's robustness against differences in simulation and reality, enabling cleaner sim-to-real transfer.

TABLE II
PPO Hyperparameters

Parameter	Value
Adam learning rate	1×10^{-4}
Adam epsilon	1×10^{-5}
discount (γ)	0.99
clipping parameter (ϵ)	0.2
epochs	3
minibatch size	64
sample size	5096

An important part of this setup is that even during training we use an *estimate* of the state rather than the true state. Though we have access to the true simulated state we instead pass the simulated sensor values into a state estimator to get a simulated "observed state." This incorporates simulated sensor noise and state estimator dynamics into the learning process, which is an essential part of making policies robust enough for sim-to-real transfer. This allows us to use the exact same controller structure on hardware, effectively just switching out the simulated robot for the real robot.

B. Learning Procedure

The reinforcement learning algorithm we use is an implementation of PPO[2] with parallelized experience collection and input normalization [14]. Our policy is a fully connected feed-forward neural network with 2 hidden layers of 256 nodes each. We choose to use fixed covariance instead of making it an additional output of the policy. The hidden layers use the ReLU activation function and the output is unbounded. This architecture was chosen because previous work found it was large and deep enough to generate high quality locomotion across a range of walking speeds [15]. More information on training hyperparameters can be found in Table II. At the start of each episode, a reference trajectory is randomly selected from the reduced-order model library and the simulated model of Cassie is set to a random starting position in the trajectory's walk cycle. A single step of agent-environment interaction includes the policy computing an action, sending it to the low-level PD controller which simulates forward 1/33 of a second, and retrieving the next state in the 33 Hz execution cycle. We define the maximum episode length for this MDP problem to be 400 steps of agent-environment interaction, which corresponds to 12 seconds. Episodes are terminated when the number of steps reaches the maximum episode length or when the reward for the current step is less than 0.3. This termination condition encapsulates when the robot falls to the ground or if it deviates excessively from the goal behavior.

We design the following reward function that is evaluated after each step:

$$\begin{aligned} r = \; & 0.3\, r_{\text{CoM\_vel}} + 0.3\, r_{\text{foot\_pos}} + 0.1\, r_{\text{straight\_diff}} \\ & + 0.2 r_{\text{foot\_orient}} + 0.1 r_{\text{action\_diff}} \end{aligned} \tag{2}$$

[1] Simulation and state estimation library available at https://github.com/osudrl/cassie-mujoco-sim

[2] Reinforcement Learning code, reward functions, and ASLIP reference motions available at https://github.com/osudrl/ASLIP-RL

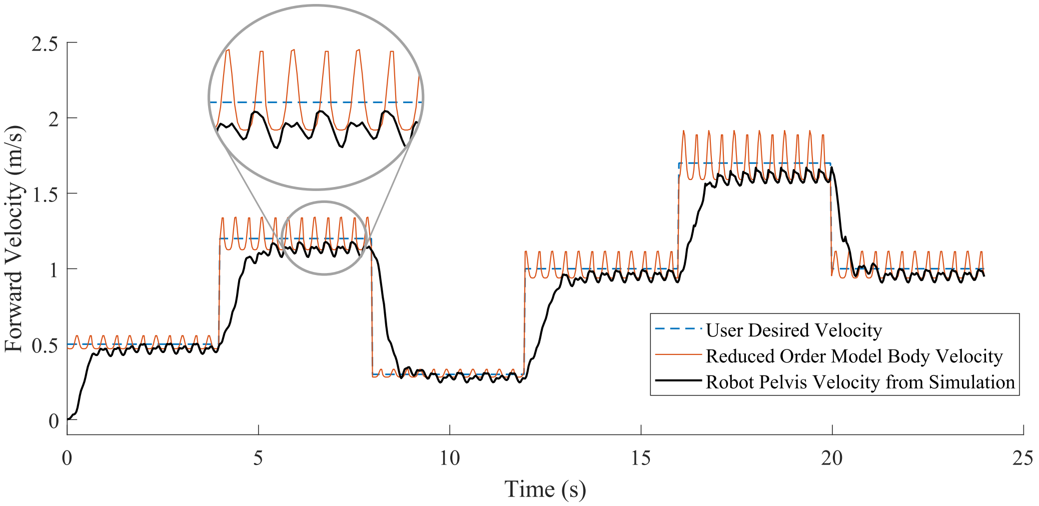

Fig. 5. Forward velocity comparison of the user desired velocity, the corresponding reduced-order model center of mass velocity, and the simulated robot's pelvis velocity. As can be seen, the policy closely tracks the user's desired velocity. The difference between the reduced-order model and robot's velocity show the learned policy does not emulate the spikes in reduced-order model's body velocity which occur at touchdown.

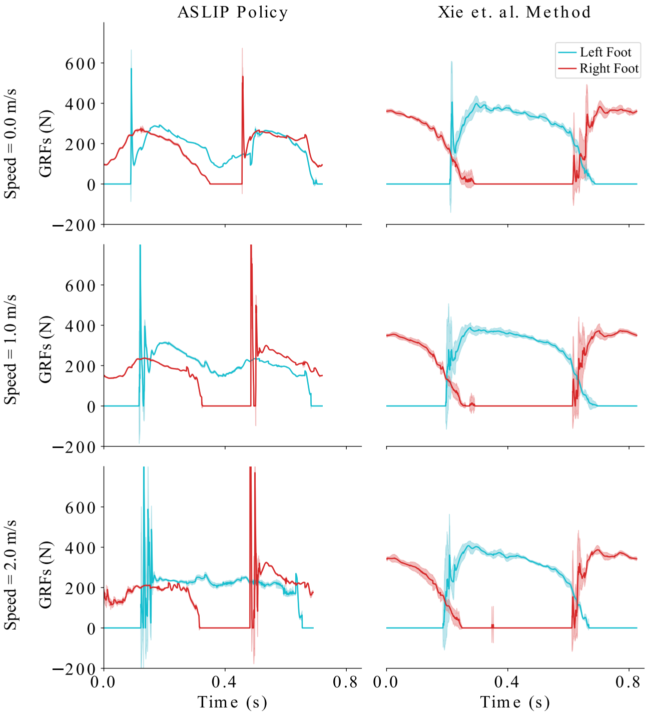

Fig. 6. Ground reaction force profiles of the actuated SLIP policy compared to a single reference trajectory policy for [15] method. The actuated SLIP policy shows double hump ground reaction forces as is expected from spring mass walking, where the single reference policy shows flattened single hump ground reaction forces similar to linear inverted pendulum walking.

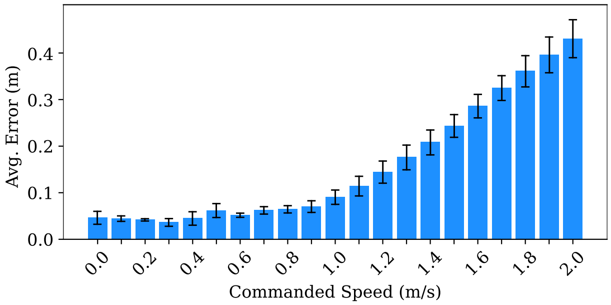

Fig. 7. Average foot placement error and standard deviation over 15 ft steps at various speeds. The average error across speeds under 1.0 m/s is relatively low and is sufficient for planning precise footstep locations.

All of the terms in the reward function are computed as the negative exponential of a distance metric. This lets us limit the maximum reward per step to 1 and per episode to 400.

The first three terms of the reward function account for 70% of the total reward. Together they penalize the differences between the current state of the robot and reference task space position and velocity. The center of mass velocity matching reward, defined as

$$r_{\text{CoM\_vel}} = \exp(-||\mathbf{v}_{\text{CoM}} - \mathbf{v}_{\text{refCoM}}||), \tag{3}$$

incentivizes matching the robot's center of mass (pelvis) velocity to the reference velocity. However, $r_{\text{CoM\_vel}}$ is calculated using the *local* pelvis frame which prevents the policy from receiving a large reward for sidestepping or walking diagonally. To ensure the robot locomotes in the forward direction, $r_{\text{straight\_diff}}$ rewards the lateral robot position being close to zero.

To get the controller to track the reference foot positions, $r_{\text{foot\_pos}}$ rewards the robot's foot positions to be close to the reference motion's foot positions, where the foot position is defined as the position of the foot relative to the body. On the full order robot, the orientation of the foot joints is important for stable walking on hardware. However, our reduced order model has point feet which do not describe or incentivize any particular foot orientation. We reward forward pointing toes and feet parallel to the ground through $r_{\text{foot\_orient}}$.

A reward term that helps the transfer to hardware is $r_{\text{action\_diff}}$, which penalizes the distance between the last action and the

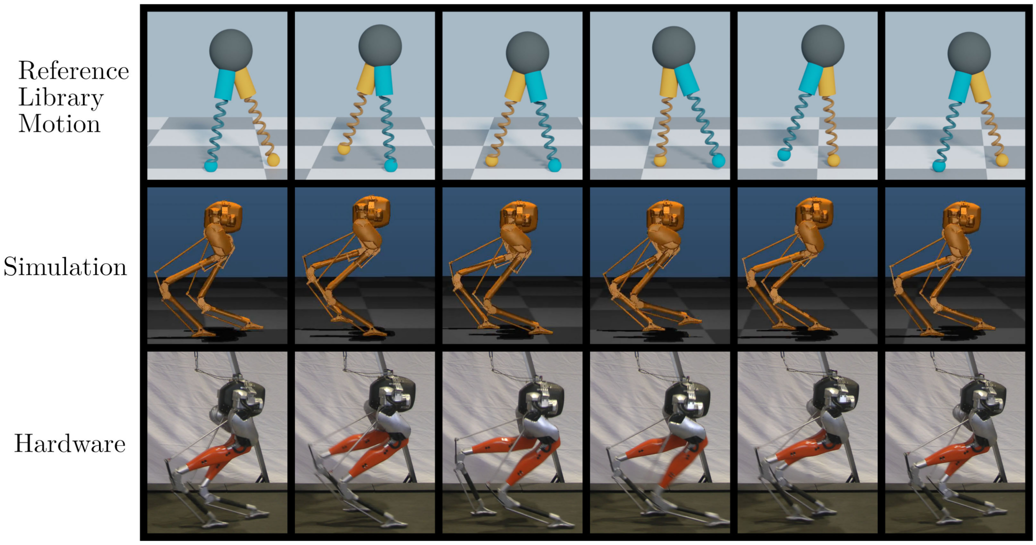

Fig. 8. Motion comparison of the reduced-order model, simulation, and hardware at different phases in the gait. The top images represent the desired reference motion to recreate, the middle images show the learned policy in simulation, and the bottom images show the learned policy executed on hardware.

current action and results in smoother action outputs. Without this term the policy can converge to behaviors that rapidly oscillate the commanded motor angles which is not conducive to success on hardware.

It is important to note that with the inclusion of the $r_{\text{action\_diff}}$ term, reaching the maximum reward is impossible and not an expectation, as the policy would need to output a constant motor angle while tracking the reference motion. Furthermore, perfectly matching the positions of the reduced-order model at each step is likely not possible because of the significantly more complex dynamics of the full order robot. As seen in [12], [13], directly applying spring-mass behavior to a robot is challenging and sensitive. Thus our learned controller should use the spring mass model as a guide towards highly effective walking solutions that work for the robot on hardware.

V. Results

We trained ten different policies from randomized initial weight seeds for our method. The training process takes just under five hours of wall clock time using 50 cores on a dual Intel Xeon Platinum 8280 server. The policy learns to step in place after about 25 million timesteps, and converges to a reward of almost 300 after about 175 million timesteps, where it is able to track all of the walk cycles in the reduced-order model library.

A. Simulation

By varying the user-provided forward velocity to the reduced-order model library, we demonstrate the learned control policy's ability to smoothly transition between discrete reference walk cycles (Fig. 5). The policy produces walking behavior with oscillations in pelvis velocity that correspond with the oscillations in center of mass velocity from the reduced-order model. Furthermore, this velocity tracking succeeds across the broad set of commanded speeds and the transitions between them.

In order to quantify how well desired foot locations can be realized through this control hierarchy, we measure the average error between the foot touchdown locations of the reduced order model commands and the robot in simulation (Fig. 7). At all speeds, we observe that the robot places its feet slightly wider than the reduced-order model. Above 1.0 m/s, we see that the robot's footsteps lag more and more which corresponds to error in tracking the reduced-order model's forward velocity. At speeds under 1.0 m/s this placement error doesn't exceed an average of 10 cm.

The ground reaction forces show that our policy produces features indicative of spring mass walking that were not explicitly incentivized by the reward function. We compare the ground reaction forces across different commanded speeds to those from the single reference trajectory policy (Fig. 6). Particularly at 0 and 1.0 m/s the actuated SLIP policy has a double hump ground reaction force which is present in spring mass walking. In comparison, the single reference trajectory has a single hump ground reaction force with a flattened peak for all speeds. This type of ground reaction force is expected from a bipedal walking policy that holds the body at a constant height, similar to the linear inverted pendulum [4]. These ground reaction forces are not themselves a useful measure of performance of the gait, but instead provide us evidence that our learned controller is emulating the dynamics of our reduced order model.

B. Hardware

We directly transfer the policies trained in simulation to hardware, demonstrating that this approach can achieve a strong sim-to-real transfer. We observe that the the learned walking motion is springy, with slight oscillations in the pelvis velocity and changes in leg length directly corresponding to the same variations in the motion of the reduced-order model. This motion correspondence can be seen in Fig. 8 and our accompanying video, which shows Cassie walking using our learned policy for an extended period of time, as well as a comparison between the motions of the reduced-order model, the learned controller in simulation, and the learned controller on hardware. The video also shows that the stepping frequency of all three stages in the control hierarchy: reduced-order model, simulation, and hardware, match for the same forward velocity commands.

To test the ability of these policies to rapidly change speeds in hardware, the human operator sends the robot sudden changes in velocity commands. The results of this trial are shown toward the end in the attached video. This controller is able to walk significantly faster and with a longer stride than was possible using previous model-based control methods on the same robot [6].

VI. Conclusion

In this letter, we have presented an effective control structure for producing spring mass-like motion on a human scale bipedal robot. This method employs reduced-order model reference trajectories to inform the learning process of the desired task space motion. We find that this method is successful in producing similar motion to the actuated SLIP model and generates policies that can realize this behavior on the bipedal robot Cassie. We found success using the actuated SLIP model as the reduced order model to guide Cassie. One should consider carefully the choice of reduced order model when applying this work to other robots. Continuations of this work will focus on extending the variety of motions the low-level policy can track and improving the foot step location tracking.

This low-level controller will enable many different opportunities for integration of high-level motion planners. Now that we have a policy capable of following a desired reduced-order model motion, we can work to extend this to generate policies that follow any arbitrary reduced-order model trajectory. This would allow for incorporating a high-level planner in the reduced-order model space, such as the planner proposed in [24]. Allowing for reactive planning decisions like navigation and obstacle avoidance to happen at the reduced-order model level will also make it significantly easier to achieve fully autonomous agile legged robots.

Acknowledgment

We would like to thank John Warila and Dylan Albertazzi for their assistance in rendering videos. Helei Duan, Jonah Siekmann, Lorzeno Bermillo, and Pedro Morais for productive discussions, and thanks to Stephen Offer, and Intel Labs for their computing resources and support.

References

[1] D. Jain, A. Iscen, and K. Caluwaerts, “Hierarchical reinforcement learning for quadruped locomotion,” 2019. [Online]. Available: https://arxiv.org/abs/1905.08926

[2] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, “Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning,” *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3699–3706, 2020.

[3] X. Da *et al.*, “Learning a contact-adaptive controller for robust, efficient legged locomotion,” 2020. [Online]. Available: https://arxiv.org/abs/2009.10019

[4] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, “The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation,” in *Proc. IEEE/RSJ Int. Conf. Intelli. Robots Syst.*, 2001, pp. 239–246.

[5] N. Motoi, T. Suzuki, and K. Ohnishi, “A bipedal locomotion planning based on virtual linear inverted pendulum mode,” *IEEE Trans. Ind. Electron.*, vol. 56, no. 1, pp. 54–61, Jan. 2009.

[6] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, “Fast online trajectory optimization for the bipedal robot cassie,” in *Proc. Robotics: Sci. Syst. XIV*, Pittsburgh, PA, USA, 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/

[7] K. Green, R. L. Hatton, and J. Hurst, “Planning for the unexpected: explicitly optimizing motions for ground uncertainty in running,” in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1445–1451.

[8] S. Heim and A. Spröwitz, “Beyond basins of attraction: Quantifying robustness of natural dynamics,” *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 939–952, Aug. 2019.

[9] J. W. Hurst, B. Morris, J. E. Chestnutt, and A. A. Rizzi, “A policy for open-loop attenuation of disturbance effects caused by uncertain ground properties in running,” in *Proc. - IEEE Int. Conf. Robot. Automat.*, 2007, pp. 1455–1460.

[10] D. E. Koditschek and M. Buhler, “Analysis of a simplified hopping robot,” *The Int. J. Robot. Res.*, vol. 10, no. 6, pp. 587–605, Dec. 1991.

[11] M. H. Raibert, *Legged Robots That Balance*. Cambridge, MA, USA: MIT press, 1986.

[12] S. Rezazadeh *et al.*, “Spring-mass walking with ATRIAS in 3D: Robust gait control spanning zero to 4.3 KPH on a heavily underactuated bipedal robot,” in *Proc. Dynamic Syst. Control Conf.*, 2015.

[13] W. C. Martin, A. Wu, and H. Geyer, “Experimental evaluation of deadbeat running on the atrias biped,” *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 1085–1092, Apr. 2017.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[15] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne, “Feedback control for cassie with deep reinforcement learning,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1241–1246.

[16] J. Tan *et al.*, “Sim-to-real: Learning agile locomotion for quadruped robots,” in *Proc. Robotics: Sci. Syst. XIV*, Pittsburgh, Pennsylvania, Jun. 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/

[17] J. Hwangbo *et al.*, “Learning agile and dynamic motor skills for legged robots,” *Sci. Robot.*, vol. 4, no. 26, 2019.

[18] M. Posa and R. Tedrake, “Direct trajectory optimization of rigid body dynamical systems through contact,” in Algorithmic Foundations of Robotics *X.*, 2013, pp. 527–542.

[19] M. S. Jones, “Optimal control of an underactuated bipedal robot,” Masters of Science in Mechanical Engineering, Oregon State University, 2014.

[20] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[21] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne, “Learning locomotion skills for cassie: Iterative design and sim-to-real,” in *Proc. 3rd Conf. Robotic Learn.*, 2019, pp. 317–329.

[22] X. B. Peng and M. van de Panne, “Learning locomotion skills using deeprl,” in *Proc. ACM SIGGRAPH / Eurographics Symp. Comput. Animation*, 2017, pp. 1-3.

[23] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 5026–5033.

[24] P. Clary, P. Morais, A. Fern, and J. Hurst, “Monte-carlo planning for agile legged locomotion,” in *Proc. Int. Conf. Automated Plan. Scheduling*, 2018, pp. 446–450.