arXiv:2011.04741v1 [cs.RO] 9 Nov 2020

Learning Task Space Actions for Bipedal Locomotion

Helei Duan[1], Jeremy Dao[1], Kevin Green[1], Taylor Apgar[2], Alan Fern[1], Jonathan Hurst[1]

***Abstract*— Recent work has demonstrated the success of reinforcement learning (RL) for training bipedal locomotion policies for real robots. This prior work, however, has focused on learning joint-coordination controllers based on an objective of following joint trajectories produced by already available controllers. As such, it is difficult to train these approaches to achieve higher-level goals of legged locomotion, such as simply specifying the desired end-effector foot movement or ground reaction forces. In this work, we propose an approach for integrating knowledge of the robot system into RL to allow for learning at the level of task space actions in terms of feet setpoints. In particular, we integrate learning a task space policy with a model-based inverse dynamics controller, which translates task space actions into joint-level controls. With this natural action space for learning locomotion, the approach is more sample efficient and produces desired task space dynamics compared to learning purely joint space actions. We demonstrate the approach in simulation and also show that the learned policies are able to transfer to the real bipedal robot Cassie. This result encourages further research towards incorporating bipedal control techniques into the structure of the learning process to enable dynamic behaviors.**

I. INTRODUCTION

Robotic legged locomotion has benefited from decades of focused analysis and rigorous control theory. Many of the most successful and capable systems choose to control actions in task space by controlling foot placements, ground reactions forces (GRFs), swing foot motions, or center-of-mass (CoM) motion to maintain balance and stability. In the simplest case, reduced order models often parameterize locomotion dynamics by choosing a particular set of actions in task space, such as feet setpoints relative to CoM along with the controlled impedance, so that the model can generate compliant behaviors and reproduce the characteristic dynamics observed from animals [1–3].

By contrast, current model-free reinforcement learning (RL) methods for controlling legged robots mostly focus on learning joint control actions and defining the reward to track joint space trajectories [4–6]. Representing learned actions in joint space may obstruct the policy from directly optimizing primary control interests and regulating task space dynamics. Joint space policies must implicitly re-learn significant information about the relation between task space and joint space in order to coordinate the joints for successful control of task space behaviors. These approaches also lack an interface to many successful model-based controllers, which exploit the task space formulation of the locomotion problems.

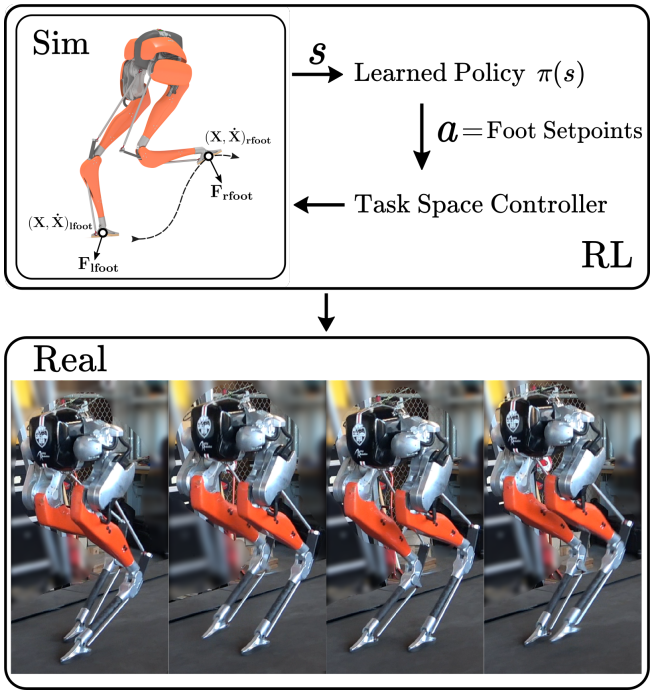


Fig. 1: Learned policy outputs actions represented in task space to directly control $\mathbf{X}, \dot{\mathbf{X}}, \mathbf{F}$. The learned policy produces the desired task space dynamics in simulation and on hardware Cassie.

Our approach aims to address such drawbacks by learning at the level of task space for legged locomotion. Actions in task space, such as feet movements, can directly influence the primary control interests and the behaviors of the agents, leading to more effective exploration than noisy, undirected exploration in joint space. Furthermore, the known models often come with the use of model-based controllers that maps from task space dynamics to joint-level commands. This allows the learned policies to interface with the same language as existing reduced order models [1] and whole-body control methods [7, 8].

In our proposed method, the learned policy outputs actions directly in task space as setpoints of the feet relative to the robot base. In particular, we integrate inverse dynamics reasoning about the known robot model so that the RL-learned task space actions are converted to joint commands. Thus, the agent does not need to learn about joint coordination and can explore directly in the more meaningful task space. In addition, the policy can now directly learn from reward functions that align with task space interests.

Our first contribution is to show that this integrated approach can significantly reduce the sample requirements

*This work is supported by the NSF Grant No. IIS-1849343, DGE-1314109, and DARPA Contract W911NF-16-1-0002.

[1]Collaborative Robotics and Intelligent Systems Institute, Oregon State University, Corvallis, Oregon, 97331, USA. Email: {duanh, daoje, greenkev, afern, jhurst@oregonstate.edu}.

[2]Agility Robotics, Albany, Oregon, 97321, USA. Email: {taylor.apgar@agilityrobotics.com}.

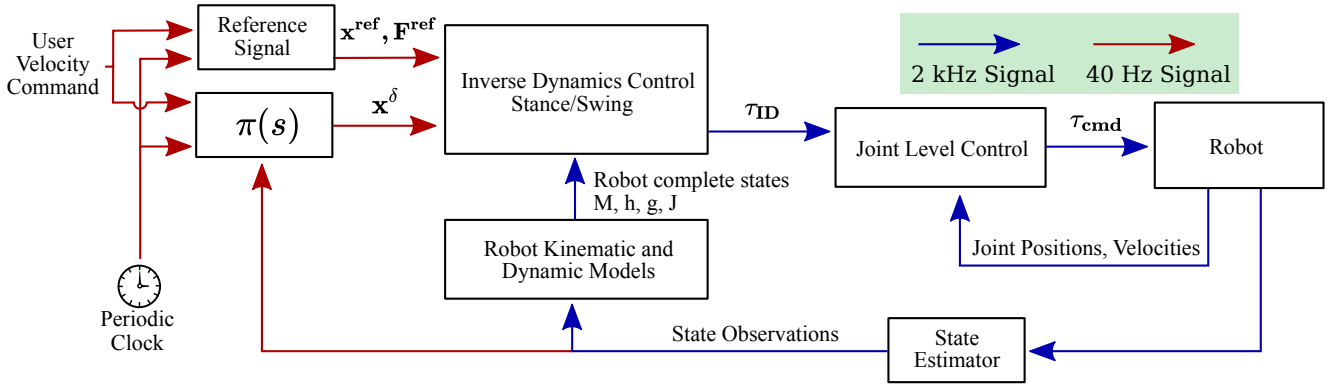


Fig. 2: Approach Overview: The output of our neural network policies $\pi(s)$ are setpoints of the feet relative to the robot base. These residual setpoint components $\mathbf{x}^{\delta} \in \mathbb{R}^{\mathbf{6}}$ together with the reference signals $\{\mathbf{x}^{\mathbf{ref}}, \mathbf{F}^{\mathbf{ref}}\}$ form the overall commands (red colored, updated at 40Hz) to the inverse dynamics control of the feet relative to the base. Furthermore, learned actions are used in different control strategies depending on the gait phase. For stance foot, the learned actions will be composed with task space impedance to produce forces that are added to the feedforward force to stabilize the robot. For swing foot, the learned actions will be converted to desired foot accelerations by task space PD control to then create compliant swing motions. Signals from the policy are commands to the inverse dynamics controller which maps commanded setpoints to torques (blue colored, 2kHz). Details are in Section III.

of reinforcement learning on bipedal locomotion, which agrees with other task space formulation in non-locomotion domains [9, 10]. Our second contribution is to demonstrate the successful steady-state walking in simulation and on hardware using the bipedal robot Cassie. We use spring mass models to generate a library of desired task space dynamics as the references for the learning process. By integrating learning with the task space controller, the learned control policies can adapt the references and produce desired control interests. To our knowledge this is the first demonstration of a task space learned policy for real-world dynamic bipedal locomotion.

II. RELATED WORK

Action Space of RL for Dynamic Robot Control. In RL problems, the action space defines the range of options agents can take to influence dynamics and make transitions between states. When using RL to generate control policies for dynamic robots, such as legged robots, the action space also defines the direct interface from policy to low-level actuator commands, such as joint positions or torques. Previous research [11] showed that using joint setpoints as the policy output into a joint PD controller provides a better interface to control fast locomotion dynamics than pure torques or joint velocities. Joint position control has then been used extensively in RL approaches for controlling legged robots in simulation and on hardware [4, 12].

Recent studies [9, 10, 13] proposed a set of new action spaces for contact-sensitive manipulation control policies by defining the action space in the manipulator's end-effector space (task space). These approaches show that, by leveraging the kinematic and dynamic models of the robot, the learning process is more sample efficient than having actions defined in joint space. The learned policy is also able to better perform contact-rich tasks with constrained dynamics on redundant and fixed-based manipulators. In our work, we extend this idea and show the sample efficient learning even when the learning method is applied to underactuated floating-base robots where contact state changes more frequently and is fundamental for system stability.

Model-based Control for Bipedal Robots. Many successful model-based controllers for bipedal robots center on task space control methods to produce desired CoM and feet dynamics. These methods are integrated closely with the known robot models that are readily available. For example, [7] uses an operational space controller to generate robust gaits while the dynamics model of the robot is used to perform fast online optimization. [14] proposes tracking the CoM trajectory and GRFs with LQR and an inverse dynamics controller. Despite many successes with model-based control methods, less is known on how to create a structured approach for learning locomotion policies that can work with existing model-based control on physical hardware. Our approach aims to provide an integrated method by bridging both sides through the same language defined in task space.

RL with Model-based Controllers. Combining conventional feedback control with reinforcement learning can be done in various ways to improve sample efficiency and take advantage of existing controllers. By decomposing actuator commands into both learned- and model-based controllers, the residual learning approach [15] has shown promise towards merging both sides. The method used in [16] improves sample efficiency by further dividing the control space in terms of uncertainty of the controller. Learning only happens near the uncertain regions where the model-based controller has less confidence. [17] proposed using RL to generate foot contact sequences that are sent to a low-level task space controller on a quadruped, which demonstrates an integrated way of generating control policies for quadrupeds using a linearized centroidal dynamics model. Our aim is to learn reactive policies in task space for a human scale bipedal robot. We formulate the low-level task space controller into swing and stance phases that takes commands from learned policies.

III. Control Structure

To facilitate the policy directly learning locomotion behaviors in task space, this section explains the task space control structure to which the learned controller issues commands. The overall control diagram is illustrated in Fig 2.

A. Task Space Constrained Dynamics

The Cassie robot's dynamics can be written in terms of its generalized, floating base coordinates. Each leg contains two passive springs and a closed kinematic loop. Following [7], we treat all of the passive spring joints as rigid for the purposes of control. We can then eliminate dependent coordinates in the leg using acceleration level loop-closure constraints [18]. In this way, the equation of motion is written in terms of only the independent coordinates so that $\mathbf{q} = [\mathbf{q_{base}} \in \mathbf{SE}(3), \mathbf{q_{joints}} \in \mathbb{R}^{10}]$ and,

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{B}\tau + \mathbf{J}^{\mathbf{T}}\mathbf{F}_{\mathbf{feet}}, \quad (1)$$

where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n\times n}$ is the inertia matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} \in \mathbb{R}^n$ includes Coriolis and centripetal forces, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ is the gravity term, and $\mathbf{B} \in \mathbb{R}^{n\times 10}$ is the actuator selector matrix given the torque at each actuated joint $\tau \in \mathbb{R}^{10}$. The Jacobian to the feet is $\mathbf{J}$ and the ground reaction force applied to the feet is $\mathbf{F_{feet}}$. The translational positions, velocities, and accelerations of the feet are a function of the generalized coordinates,

$$\begin{aligned}\mathbf{x} &= \mathbf{FK}(\mathbf{q})\\ \dot{\mathbf{x}} &= \mathbf{J}\dot{\mathbf{q}}\\ \ddot{\mathbf{x}} &= \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}},\end{aligned}$$

where $\mathbf{FK}(\mathbf{q})$ is forward kinematics of the legs.

B. Task Space Control

The task space inverse dynamics controllers consist of swing phase, stance phase, and the transition between the two. This basic setup allows the learning to explore in terms of different subfunctions in locomotion. $\mathbf{x}, \mathbf{x}^{\delta}, \mathbf{x^{ref}}$ are feet positions from the state estimator, the learned policy, and references.

1) Swing leg: Swing leg control is formed as a PD acceleration controller with inverse dynamics on feet to reach to desired setpoints relative to base. First, the desired feet accelerations based on task space PD feedback is calculated as,

$$\ddot{\mathbf{x}}^{\mathbf{des}} = \mathbf{K_P^{swing}}(\mathbf{x^{ref}} + \mathbf{x}^{\delta} - \mathbf{x}) + \mathbf{K_D^{swing}}(\mathbf{0} - \dot{\mathbf{x}}).$$

Then we can compute the motor torques required to create the commanded task space acceleration using task space inertia, $(\mathbf{\Lambda} = (\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathbf{T}})^{-1})$. We dropped the velocity product terms from (1) and $\dot{\mathbf{J}}\dot{\mathbf{q}}$ resulting in,

$$\tau_{\mathbf{swing}} = \mathbf{J}^{\mathbf{T}}(\mathbf{\Lambda}\ddot{\mathbf{x}}^{\mathbf{des}} + \mathbf{\Lambda}\mathbf{J}\mathbf{M}^{-1}\mathbf{G}).$$

2) Stance Leg: The stance leg is primarily concerned with force interactions, so the control converts the set point commands into forces though the impedance controller with feedforward forces,

$$\mathbf{F_{feet}^{des}} = \mathbf{K_P^{stance}}(\mathbf{x^{ref}} + \mathbf{x}^{\delta} - \mathbf{x}) + \mathbf{K_D^{stance}}(\mathbf{0} - \dot{\mathbf{x}}) + \mathbf{F^{ref}}.$$

This force command is translated into joint torques through the simple Jacobian transpose control of feet to robot base,

$$\tau_{\mathbf{stance}} = \mathbf{J}^{\mathbf{T}}\mathbf{F_{feet}^{des}}.$$

This is the torque that would be required to statically apply the force command in the absence of all other forces.

3) Transitions between Stance/Swing: The control structure also includes a transitioning rule to switch between the two phases of the gait, and relies on RL to find solutions to stabilize the robot. For each leg, the overall torque computation is,

$$\tau_{\mathbf{ID}} = \phi\tau_{\mathbf{stance}} + (1-\phi)\tau_{\mathbf{swing}}, \quad (2)$$

where $\phi \in [0, 1]$ is a scalar output that depends on the phase along the target force profile on each leg, meaning if the target force profile is active, then $\phi = 1$ and if it is not active, then $\phi = 0$. This also allows for a double support phase where ϕ smoothly transits between stance and swing phases. During the transition phases around toe-off and touchdown, ϕ linearly interpolates between 0 and 1 and vice versa.

4) Feet Orientation Control: Swing and stance control only commands the translational positions of the feet. Foot orientation can be controlled via joint control of the hip yaw and foot pitch motors on each leg $\theta = \{\theta_{\mathbf{hipyaw}}, \theta_{\mathbf{footpitch}}\}$. We use joint PD controllers on only the hip yaw and foot pitch motors. The policy outputs θ^{δ} to control the feet orientation to be flat and facing forward. $\theta^{\mathbf{ref}}$ represents a neutral joint position and is constant for each motor. These complimentary torques are defined as,

$$\begin{aligned}\tau_{\mathbf{orient}} &= \{\tau_{\mathbf{hipyaw}}, \tau_{\mathbf{footpitch}}\}\\ &= \mathbf{K_P^{joint}}(\theta^{\mathbf{ref}} + \theta^{\delta} - \theta) - \mathbf{K_D^{joint}}\dot{\theta}.\end{aligned} \quad (3)$$

5) Joint Level Commands and Control Rate: The final torque commands to low-level actuators are,

$$\tau_{\mathbf{cmd}} = \tau_{\mathbf{ID}} + \tau_{\mathbf{orient}}. \quad (4)$$

As shown in Fig. 2, the full inverse dynamics control runs at 2kHz to compute torques based on (4). The RL policy outputs residual setpoints at 40Hz, meaning the desired target setpoints are updated at 40Hz.

IV. RL Problem Formulation

A. Learning Algorithm

We use the Proximal Policy Optimization (PPO) algorithm [19], a model-free policy gradient method with an actor-critic network. Both the actor and the critic use fully connected feed-forward neural networks with one hidden layer of size 256 and ReLU activation. Output layers use $tanh$ so that the policy outputs are bounded. The hyperparameters for the algorithm are listed in the Appendix I.

B. Policy Inputs

The input to the neural network policy consists of the observed robot state and user commands. The entire observations $\mathbf{S} \in \mathbb{R}^{47}$ input to the neural network are measured through the proprietary state-estimator from Agility robotics. Using estimated states as inputs allows sim-to-real transfer without extra steps when considering possible differences between simulator and hardware. The state space covers primarily the observations on the robot base and feet. More specifically, $\mathbf{S}$ includes commanded forward velocity $\mathbb{R}^1$, base's translational, rotational velocities $\mathbb{R}^6$ and orientations in quaternion $\mathbb{R}^4$, actuated and un-actuated joint positions and velocities measurable on hardware $\mathbb{R}^{28}$, and feet Cartesian positions relative to robot base $\mathbb{R}^6$. Also, $\mathbf{S}$ has an additional $\mathbb{R}^2$ input based on the phase ϕ variable along the references. This input is calculated with both sine and cosine functions to create unique phase identifiers and to wrap around the cycle.

C. Action Space

The action space $\mathbf{A} \in \mathbb{R}^{10}$ is defined as residual terms, including the Cartesian positions of feet $\mathbb{R}^6$ relative to the robot floating-base and the joint positions of hip yaw and foot pitch $\mathbb{R}^4$ for foot orientation control in (3).

D. Reward

The reward centers on achieving task space control interests from the references for the robot base and end-effector feet. The desired reference quantities are defined with the corresponding phase along the references. We chose to not fully track the references because a tracking-only reward could result in the policy fitting too much to trajectories that do not have the notion of the full-order robot, making performance very brittle. Rather, we define a more general reward based on the fundamentals of locomotion: transitions between stance and swing. We still use the information encoded by the references by having them define the contact sequence and timing, but this method allows for more adaptive solutions.

Rewards on the feet r_{lfoot} or r_{rfoot} are defined as locomotion subfunctions, swing and stance for each foot. Swing reward $r_{\text{foot-swing}}$ calculates the foot clearance cost weighted by swing foot velocities. Stance reward $r_{\text{foot-stance}}$ checks whether the foot is in contact with the ground by calculating both foot height and velocities. With the references from reduced-order models, the timing of swing and stance is predefined, as is the desired foot clearance height $h_{\text{swing-height}}$ which is set to 0.15m. Similar to transitions defined in controller (4), a continuous transition weighting $\phi \in [0, 1]$ indicates whether each foot should be in stance or swing,

$$r_{\text{lfoot}} \text{ or } r_{\text{rfoot}} = \phi r_{\text{foot-stance}} + (1 - \phi) r_{\text{foot-swing}}.$$

The reward function then consists of all the control interests on the robot floating-base defined by references on $r_{\text{base-xvel}}$, $r_{\text{base-zvel}}$, and $r_{\text{base-zpos}}$, along with r_{stable} to encourage smooth, stable behaviors as,

$$\begin{aligned} \mathrm{R} = {} & 0.1 r_{\text{lfoot}} + 0.1 r_{\text{rfoot}} + 0.15 r_{\text{base-xvel}} + 0.1 r_{\text{base-zvel}} \\ & + 0.1 r_{\text{base-zpos}} + r_{\text{stable}}, \end{aligned} \tag{5}$$

Reward details are in Appendix II.

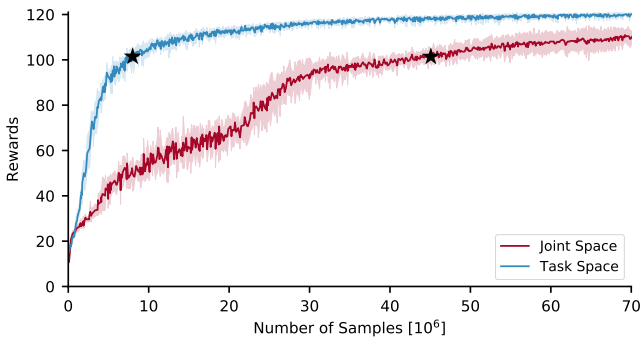


Fig. 3: We trained five policies for each action space with random seeds. Actions in task space are able to learn rough stepping gaits at the beginning of the training and quickly converge to a plateau. However, actions in joint space need to explore for joint coupling before the agents can effectively explore in a more useful episode. ★ represents when the robot starts to walk stably in simulation with the learned policy, which takes around 8 and 32 hours (wall-clock time) for task space and joint space respectively.

V. Results

A. Experiment Setup

For our experiments, the learning problem is given with a set of reference signals $\mathbf{x}^{\mathbf{ref}}, \mathbf{F}^{\mathbf{ref}}$ that are optimized based on the actuated spring mass models in the extended work from [1]. However, this library of profiles cannot be directly executed on the robot to walk and thus used as feedforward signals in the inverse dynamics controller. The gains for the task space controllers are $\mathbf{K_P^{swing}} = \{300, 150, 400\}\text{s}^{-2}$ and $\mathbf{K_D^{swing}} = \{10, 3, 10\}\text{s}^{-1}$ for the forward/lateral/vertical directions respectively. $\mathbf{K_P^{stance}}$ and $\mathbf{K_D^{stance}}$ have the same corresponding values but with units N/m and Ns/m. The gains for the hip yaw motors and the foot pitch motors are $\mathbf{K_P^{joint}}$ ={100, 50}Nm/rad and $\mathbf{K_D^{joint}}$ ={10, 5}Nm/rads^{-1} respectively. The gains are the same in simulation and on hardware.

For the sampling and training process, we use *cassie-mujoco-sim* simulation packages[1] based on the Mujoco physics simulator [20]. The training horizon for each episode is 150 steps, which corresponds to running the policy at 40Hz for 3.75 seconds. We sample 5000 simulation timesteps for each iteration of training. During the learning process, the initial conditions of the robot are randomized for each episode, including the reference phase and the velocities of the robot base to simulate disturbances. The termination condition of each episode only considers if the robot base height falls below 0.6 meters. Policies were trained using 50 cores on a dual Intel Xeon Platinum 8280 server.

To compare the sample efficiency and task space dynamics with actions learned in joint space, we implemented a joint

[1]https://github.com/osudrl/cassie-mujoco-sim

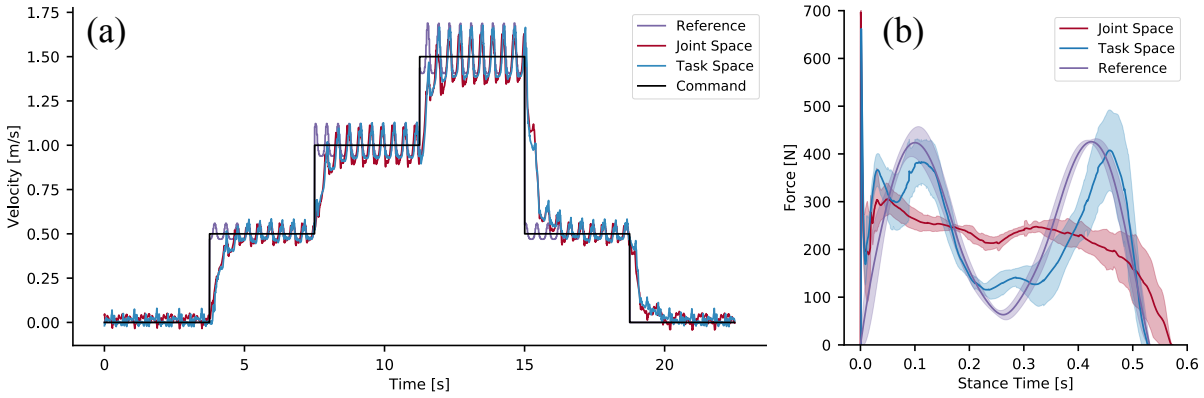


Fig. 4: Simulation results for learned policies. Plot (a) shows a speed tracking comparison. Plot (b) shows the ground reaction forces recorded from the speed tracking test averaged over the stance period.

space action learning approach similar to previous work [4]. To make both action spaces equivalent, the joint space references are numerically derived using offline inverse kinematics on the same set of reference motions. Also, the hyperparameters and other learning setup is equivalent on both action spaces.

B. Simulation Results

We primarily focused on comparisons in simulations for sample efficiency and how well each action space achieves the desired task space control interests.

1) Sample Efficiency: We trained policies for both action spaces to compare sample efficiency under the same reward and learning setup. Fig. 3 shows that learning task space actions reaches high reward much faster than learning joint space actions. During the early stages of training, actions in task space have a direct effect on the robot dynamics by moving the feet around in coordination, as opposed to actions in joint space which results in the robot standing still without any intentions to move the robot base towards the desired velocities. We show the during-training comparison in the video and discuss the potential cause further in Section V-C.

2) Commanded Velocity Tracking: We tested the steady-state locomotion by sending varying desired speed commands into the corresponding input to the policy. During training, desired speed for each episode holds constant, but the robot is initialized with random velocity of the robot base. The final learned policy can interpolate across the desired speed range as shown in Fig. 4(a). Both action spaces have comparable performance on speed tracking.

3) Ground Reaction Forces: We used the optimized reduced-order references to derive the references in task space and joint space. By following the desired CoM dynamics defined in the reward, the ground reaction forces should capture the trend described from the references as seen from Fig. 4(b). Ground reaction forces are recorded during the speed tracking experiments where Cassie walks across the speed range. Although policies by both action spaces are able to produce stable locomotion behaviors, the underlying ground reaction forces are very different. Directly learning task space actions produces the desired ground reaction forces over repeated gait cycles, whereas joint space actions cannot reproduce the desired stance dynamics even after reward plateaus. Furthermore, we can see that the task space actions are not exactly following the feedforward force profile, but rather adapting the profile and stretching the mid to late stance to achieve better stability.

C. Exploration Noise in Action Space

Noise in learning algorithms allows agents to explore their environment but it can make learning very brittle without the proper tuning. For legged locomotion tasks, the robot dynamics are primarily influenced through the motion of the feet. In order to evaluate the effective exploration by both action spaces, we compare the range of task space explored. In our experiments we find that the exploration noise σ in the Gaussian policies must be different for each action space in order to produce feasible policies. Thus, we analyzed the effects of the noise for each action space independent of the particular learning algorithms. Fig. 5 shows the comparison of effective task space motions sampled over each action space. Actions are drawn from a Gaussian distribution $a \sim \mathcal{N}(0, \sigma^2)$, where $log\sigma = -1.5$ for joint space and $log\sigma = -2.5$ for task space, same as used in PPO in Appendix I. In the first scenario, data is collected using single step sampling where the simulation is terminated and reset after every single step. The second scenario collects samples in the episode setting, where random actions are continually drawn and acted until a termination condition is satisfied or the horizon is reached. We collected 50,000 samples for each scenario.

Though there is little difference in the single step scenario, task space sampling covers a much larger area of the state space in the episode scenario. Furthermore, since termination conditions are used, the data that task space sampling generates captures dynamics before falling. In other words, it is not just the case that task space sampling generates a larger range of states, it generates a larger range of *useful* states. We noticed that simply increasing the noise in joint space to $log\sigma = -1$ during sampling did not increase the feet range of motions, because the extra noise would just cause the robot to fall down and lead to bad states cutoff by the termination condition. This result highlights the need

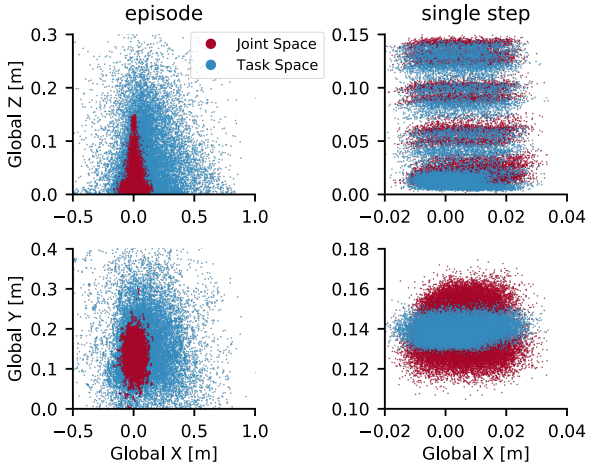


Fig. 5: Difference in the random exploration happens at the episodic sampling, where task space shows a more direct exploration over entire episode that results into a large space for feet movements, and joint space only covers a small space. "Stripe-like" samples are a result of using a small set of initialization states of the robot.

for effective *directed* exploration for legged locomotion in its natural action space. One cannot just use more random noises in an effort to sample more states to improve learning. Task space both contains the primary control interests and allows for directed exploration that greatly improves sample efficiency.

D. Hardware Transfer

We are able to transfer the learned policies onto Cassie without additional randomizations on robot model parameters. The supplementary video shows the robot is capable of stepping in-place and walking at various speeds on treadmill.

VI. CONCLUSION

In this work, we present an approach that combines bipedal robot control with reinforcement learning to enable a policy to learn actions in task space. We show that learning actions in task space can produce steady-state locomotion behaviors and demonstrated the successful sim-to-real transfer on Cassie. Specifically, the learned actions represent setpoint residuals relative to the robot base. By varying this setpoint over the stance and swing phases, the learned policies adapt the references trajectories while being able to reject disturbances.

The proposed approach unloads RL from having to re-learn the known robot models, so it is sample efficient and explores the task space regions more broadly to find solutions. We found that simply increasing the action noises in joint space does not correlate to better exploration, because undirected exploration in Gaussian policies may gather many useless samples. On the other hand, directly giving such joint coordination information into learning may hinder the discovery of interesting coupling effects among the joints. It is possible that learning to approximate the joint-to-task space translation will find solutions that use the robot dynamics in interesting ways. However, one can learn this translation independently so that the exploration can still be done in task space with more directed and informative samples. Further research may focus on how to embed the notions of the effective task space changes into the learning process.

Learning task space action could also allow RL to communicate in the same language with potentially various layers of a control hierarchy, where exchange signals are mostly defined in task space. This could be a necessary step to bridge reinforcement learning with many existing control methods. Future research may benefit on incorporating more advanced model-based control designs, such as adding centroidal dynamics as one of the actions, to design structured learning approach for more dynamic and agile locomotion behaviors on real hardwares.

APPENDIX

A1. Hyperparameters for Proximal Policy Gradient implementations,

Parameter	Value
Horizon (H)	150
Adam stepsize	1×10^{-4}
Number epochs	8
Minibatch size	1024
Discount (γ)	0.99
Clipping parameter (ϵ)	0.2
Max gradient norm	0.05
Log stddev. of action distribution	-2.5 (task space)
	-1.5 (joint space)

TABLE I: PPO Hyperparameters

A2. Each reward term is a function of one or more physical quantities of the robot s. And $r(s) = \exp\left(\frac{1}{T}\sum_{i=0}^{T} f(s)\right)$ uses an exponential as the kernel function to normalize the values f to create smooth reward between $(0, 1]$ for each reward term. Each f is averaged over the duration $T = 50$ between each policy update, given torque control frequency. The neutral orientation in quaternion $\mathbf{q}_{\text{neutral}} = (1, 0, 0, 0)$.

Terms (f)	Expressions
base-xvel	$3\lvert\dot{x}_{\text{base}} - \dot{x}_{\text{base-ref}}\rvert$
base-zvel	$3\lvert\dot{z}_{\text{base}} - \dot{z}_{\text{base-ref}}\rvert$
base-zpos	$3\lvert z_{\text{base}} - z_{\text{base-ref}}\rvert$
foot-stance	$x^2_{\text{foot-height}} + \lVert\dot{\mathbf{x}}_{\text{foot}}\rVert_2$
foot-swing	$40(h_{\text{foot-height}} - x_{\text{z-height}})^2$
base-linear-accel	$\lVert\ddot{\mathbf{x}}_{\text{base}}\rVert_2$
base-angular-vel	$\lVert\boldsymbol{\omega}_{\text{base}}\rVert_2$
base-orientation	$50(1 - \langle\mathbf{q}_{\text{base}}, \mathbf{q}_{\text{neutral}}\rangle)$
base-straight	$5\lvert y_{\text{base}}\rvert + 3\lvert\dot{y}_{\text{base}}\rvert$
base-orientation	$50(1 - \langle\mathbf{q}_{\text{base}}, \mathbf{q}_{\text{neutral}}\rangle)$
foot-orientation	$30(1 - \langle\mathbf{q}_{\text{foot}}, \mathbf{q}_{\text{neutral}}\rangle)$
action-smooth	$3\lVert\mathbf{a_i} - \mathbf{a_{i-1}}\rVert_2$
r_{stable}	$0.1r_{\text{base-orientation}} + 0.15r_{\text{base-straight}}$
	$+0.05r_{\text{foot-orientation}} + 0.025r_{\text{base-linear-accel}}$
	$+0.025r_{\text{base-angular-vel}}$
	$+0.1r_{\text{action-smooth}}$

TABLE II: Reward Function Details

ACKNOWLEDGMENTS

We thank Intel for providing vLab resources. We also thank DRL members and Jianwen Luo for helpful discussions.

REFERENCES

[1] K. Green, R. L. Hatton, and J. Hurst, "Planning for the unexpected: Explicitly optimizing motions for ground uncertainty in running," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1445–1451.

[2] H. Geyer, A. Seyfarth, and R. Blickhan, "Compliant leg behaviour explains basic dynamics of walking and running," *Proceedings of the Royal Society B: Biological Sciences*, vol. 273, no. 1603, pp. 2861–2867, 2006.

[3] Y. Blum, H. R. Vejdani, A. V. Birn-Jeffery, C. M. Hubicki, J. W. Hurst, and M. A. Daley, "Swing-leg trajectory of running guinea fowl suggests task-level priority of force regulation rather than disturbance rejection," *PLoS ONE*, vol. 9, no. 6, pp. 18–20, 2014.

[4] 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," *Conference on Robotic Learning*, no. CoRL, 2019.

[5] 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," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[6] J. Siekmann, S. Valluri, J. Dao, F. Bermillo, H. Duan, A. Fern, and J. Hurst, "Learning Memory-Based Control for Human-Scale Bipedal Locomotion," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[7] T. Apgar, P. Clary, K. Green, A. Fern, and J. Hurst, "Fast Online Trajectory Optimization for the Bipedal Robot Cassie," *Robotics: Science and Systems XIV*, 2018. [Online]. Available: http://www.roboticsproceedings.org/rss14/p54.pdf

[8] J. Luo, Y. Zhao, D. Kim, O. Khatib, and L. Sentis, "Locomotion control of three dimensional passive-foot biped robot based on whole body operational space framework," *2017 IEEE International Conference on Robotics and Biomimetics, ROBIO 2017*, vol. 2018-Janua, no. December, pp. 1577–1582, 2018.

[9] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1010–1017.

[10] P. Varin, L. Grossman, and S. Kuindersma, "A comparison of action spaces for learning manipulation tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6015–6021.

[11] X. B. Peng and M. van de Panne, "Learning locomotion skills using deep RL: Does the choice of action space maer?" *Proceedings - SCA 2017: ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, no. 1, 2017.

[12] 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, pp. 1–20, 2019.

[13] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3080–3087.

[14] W. C. Martin, A. Wu, and H. Geyer, "Experimental evaluation of deadbeat running on the ATRIAS biped," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1085–1092, 2017.

[15] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, pp. 6023–6029, 2019.

[16] M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox, "Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 7505–7512.

[17] X. Da, Z. Xie, D. Hoeller, B. Boots, A. Anandkumar, Y. Zhu, B. Babich, and A. Garg, "Learning a Contact-Adaptive Controller for Robust, Efficient Legged Locomotion," no. CoRL, 2020. [Online]. Available: http://arxiv.org/abs/2009.10019

[18] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," pp. 1–12, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[20] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," *IEEE International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.