Reinforcement Learning for Robust Parameterized Locomotion Control of Bipedal Robots

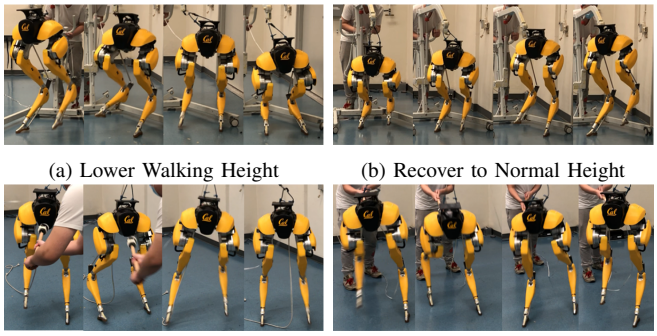
Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, Koushil Sreenath

***Abstract*— Developing robust walking controllers for bipedal robots is a challenging endeavor. Traditional model-based locomotion controllers require simplifying assumptions and careful modelling; any small errors can result in unstable control. To address these challenges for bipedal locomotion, we present a model-free reinforcement learning framework for training robust locomotion policies in simulation, which can then be transferred to a real bipedal Cassie robot. To facilitate sim-to-real transfer, domain randomization is used to encourage the policies to learn behaviors that are robust across variations in system dynamics. The learned policies enable Cassie to perform a set of diverse and dynamic behaviors, while also being more robust than traditional controllers and prior learning-based methods that use residual control. We demonstrate this on versatile walking behaviors such as tracking a target walking velocity, walking height, and turning yaw. (Video[1])**

I. Introduction

Many environments, particularly those designed for humans, are more accessible by legged systems. However, bipedal robot locomotion involves several control design challenges due to high degrees-of-freedom (DoFs), hybrid nonlinear dynamics, and persistent but hard-to-model ground impacts. Classical model-based methods [1]–[3] for stabilizing and controlling bipedal systems tend to require careful modeling and usually lack the ability to adapt to changes in the environment. Recent deep reinforcement learning (RL) based methods are promising solutions to these issues as RL is able to leverage the full-order dynamics of the system to produce more agile behaviors.

Our approach to bipedal locomotion utilizes RL to train robust policies to imitate gaits from a gait library, using randomized simulated training to acquire controllers that can successfully control a person-sized bipedal robot Cassie in the real world. Recent RL methods on Cassie [4], [5] train policies to specify corrections to reference motions recorded from a model-based walking controller. While this type of residual control [6] can produce stable walking behaviors, it often requires a pre-existing controller, and the resulting behaviors tend to be limited to stay close to the original reference motions. To overcome this limitation, our training system uses a gait library of diverse parameterized motions based on Hybrid Zero Dynamics (HZD) [3], which increases the diversity of behaviors that the robot can learn. This increase in diversity serves two purposes. First, it enables the online control of the robot using a low dimensional parameterization of different gaits. Second, it improves robustness by enabling the robot to learn a larger variety of potential motions.



(a) Lower Walking Height (b) Recover to Normal Height

(c) Push Recovery (Front) (d) Push Recovery (Back)

Fig. 1: Our system leverages reinforcement learning to train robust parameterized locomotion controllers for a bipedal Cassie robot. The controller can vary parameters such as walking velocity and height, while also being robust to significant external perturbations.

Due to the size and instability of real bipedal robots, it is particularly dangerous to perform RL directly on the physical system. We instead leverage techniques from sim-to-real transfer to train policies in a simulated environment, which are then evaluated in another higher fidelity simulator, before finally being deployed on the real robot. We found that additional dynamics randomization techniques are necessary to produce robust controllers, and these enable our system to learn robust parameterized controllers, which are then evaluated on Cassie with a collection of different motions and challenging scenarios, as shown in Fig. 1.

The primary contribution of this work is the development of a reinforcement learning based controller as shown in Fig. 2 that results in a more diverse and robust walking control on the Cassie robot. More specifically, we develop an end-to-end versatile walking policy that combines a HZD-based gait library with deep reinforcement learning to enable a $3D$ bipedal robot Cassie to walk while following commands for frontal and lateral walking speeds, walking height, and turning yaw rate. The proposed learning-based walking policy notably expands the feasible command set and safe set over prior model-based controllers, and improves stability during gait transitions compared to a HZD-based baseline walking controller. The learned policies are robust to modelling error, perturbations, and environmental changes. This robustness emerges from our training strategy, which

All authors are with the University of California, Berkeley, CA, USA. {zhongyu_li, chengxuxin, xbpeng, gberseth}@berkeley.edu, pabbeel@cs.berkeley.edu, svlevine@eecs.berkeley.edu, koushils@berkeley.edu

This work is supported in part by National Science Foundation Grants CMMI-1944722 and IIS-1651843, the Office of Naval Research, NSERC Postgraduate Scholarship, a Berkeley Fellowship for Graduate Study, and BAIR.

[1]Video: https://youtu.be/goxCjGPQH7U

trains a policy to imitate a collection of diverse gaits, and also incorporates domain randomization. The learned policy can be directly transferred to other simulators, such as a more accurate simulator on SimMechanics, as well as to a real robot. Using this proposed policy, Cassie is not only able to reliably track given commands in indoor and outdoor environments, but to stay robust to unmodelable malfunctioning motors, changes of ground friction, carrying unknown loads, and demonstrating agile recoveries from random perturbations, as shown in the video.

A. Related Work

Traditional approaches for locomotion of bipedal walking robots are typically based on *notions of gait stability*, such as the ZMP criterion [1] and capturability [2], simplified models [7]–[9], and constrained optimization methods [10]–[12]. These methods have been shown to be effective for controlling various humanoid robots with flat feet, but the resulting motion tends to be slow and conservative. Hybrid Zero Dynamics (HZD) [3], [13]–[17] is another control technique for generating stable periodic walking gaits based on input-output linearization. Our work, which is based on reinforcement learning, is not constrained by the requirement of a precise model and stabilization to a periodic orbit, as is the case for HZD, which enables our method to produce more diverse behaviors.

a) RL-based Control for Legged Robots: Reinforcement learning for legged locomotion has shown promising results in acquiring locomotion skills in simulation [18]–[20] and in the real world [21]–[23]. Data-driven methods provide a general framework that enables legged robots to perform a rich variety of behaviors by introducing reference motion terms into the learning process [20], [23], [24]. However, most previous RL-based work are deployed on either multi-legged systems [23], [25], [26] or on low-dimensional bipedal robots [27], [28], where learned motions are typically quasi-static.

More recently, RL has been applied to learn agile walking skills for Cassie. Model-based RL in [29], [30] attains a velocity regulating adaptive walking controller on Cassie in simulation. In [4], [5], reference motions combined with model-free residual learning [6] are used to learn walking policies that are able to reliably track given planar velocity commands on Cassie in the real world. Residual control structure used in the policy can speed up training, but the resulting policy can only apply limited corrections to the underlying reference trajectory. Moreover, a model-based walking controller on Cassie is still needed to provide reference motions recorded from control outputs. This limits the accessibility to the reference motions and therefore reduces the diversity of learned behaviors. In addition, most of the previous learning-based walking policies on Cassie do not show significant improvement over traditional model-based controllers. Also, they lack the ability to change the walking height and turning yaw, which increases the complexity of controller design but enables the robot to travel in narrow environments. In our work, we show a clear improvement

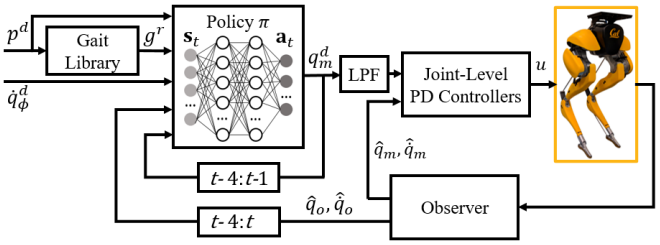


Fig. 2: Proposed learning-based walking controller. The inputs of the policy consists of desired gait parameter p^d, desired turning yaw velocity $\dot{q}^d_\phi$, a reference gait g^r decoded by desired gait parameter p^d, observed robot states $\hat{q}_o$, $\hat{\dot{q}}_o$ from time step t-4 to t, and past policy outputs which are desired motor positions q^d_m spanning from t-4 to t-1. Current q^d_m is sent to joint-level controllers after passing through a Low Pass Filter (LPF).

over model-based methods by examining the tracking performance and robustness over a range of gait parameters.

b) Simulation to Real World Transfer: Sim-to-real transfer is an attractive approach for developing policies, which takes advantage of fast simulations as a safe and inexpensive source of data. Model-based methods require careful system identification to bridge the reality gap [16], [17]. Randomizing the system properties in the source domain in order to cover the uncertainty in the target domain has allowed for solutions that use low-fidelity simulations for learning-based methods [23], [28], [31]–[35]. In this paper, we adopt domain randomization to overcome the sim-to-real gap, without the need for any additional training on the robot.

II. Parameterized Control of Cassie

We now present the Cassie bipedal robot, which is the platform for our experiments and introduce a HZD-based gait library of versatile walking behaviors on Cassie.

A. Cassie Robot Model

Cassie is a person-sized, dynamic, underactuated bipedal robot with 20 DoFs, as shown in Fig. 1 and explained in [17, Sec. II]. There are 10 actuated rotational joints $q_m = [q^{L/R}_{1,2,3,4,7}]^T$, which include abduction, rotation, hip pitch, knee, and toe motors. There are also four passive joints $q^{L/R}_{5,6}$ that correspond to the shin and tarsus joints. Its floating base pelvis $q_p = [q_x, q_y, q_z, q_\psi, q_\theta, q_\phi]^T$ has 3 transitional DoFs (sagittal, lateral, vertical) $q_{x,y,z}$ and 3 rotational DoFs (roll, pitch, yaw) $q_{\psi,\theta,\phi}$, and is defined as the robot's local reference frame. The full robot state $q \in \mathbb{R}^{20}$ consists of the state of each joint. Next, we define the observable state $q^o \in \mathbb{R}^{17}$, which is similar to q but excludes the pelvis translational position $q_{x,y,z}$, that can not be reliably measured in the real world without external instrumentation.

B. Gait Library and Parameterized Control

To create a controller that can be directed online to perform and transition between different motions, we parameterize the input to the system using a *gait parameter* p that determines the desired *gait*. A *gait* g is a set of periodic joint trajectories that encode a locomotion behavior [36]. In this work, we use 5^{th} order Bézier curves α to represent

TABLE I: Gait Library

	$\dot{q}_x$	$\dot{q}_y$
$\mathcal{G}$	$[-1, -0.8, \ldots, 0.8, 1.0]$	$[-0.3, -0.24, \ldots, 0.24, 0.3]$
	q_z	number of gaits
$\mathcal{G}$	$[0.65, 0.685, \ldots, 0.965, 1.0]$	$11 \times 11 \times 11 = 1331$

smooth profiles for the 10 actuated joints. The Bézier curves are normalized by 1 step period $\bar{t}$. The gaits designed in this paper consist of 2 steps, referred to as *right stance* and *left stance*, and transitions between the steps are triggered by a foot *impact* on the ground. The gait parameters chosen in this work are forward velocity $\dot{q}_x$, lateral velocity $\dot{q}_y$ and walking height q_z, *i.e.*, $p = [\dot{q}_x \ \dot{q}_y \ q_z]^T \in \mathbb{R}^3$. A *gait library* $\mathcal{G} = \{g_i(\alpha, \bar{t})\}$ is constructed by indexing the i^{th} gait g_i with its gait parameter p_i. The optimization program for constructing the HZD-based gait library is formulated in CFROST [13] and the resulting gaits are described in Tab. I [17]. The gait libray is later combined with an online regulator to implement a parameterized walking controller in [17, Sec. IV].

III. Learning Walking Control and Sim-to-Real

Having an optimized gait library is not enough to control bipedal robots without online feedback, we will next combine the pre-computed HZD-based gait library with reinforcement learning to develop a versatile locomotion policy π for the Cassie. In a RL framework, an agent (*e.g.* Cassie), learns through trial and error by interacting with the environment. At each discrete time step t, the policy π (shown in Fig. 2) observes a state $\mathbf{s}_t$ and a goal $\mathbf{g}_t$, and outputs an action distribution $\pi(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g}_t)$. The agent then samples an action $\mathbf{a}_t$ from the distribution and executes the action in the environment, which results in a transition to a new state $\mathbf{s}_{t+1}$ and goal $\mathbf{g}_{t+1}$, as well as a reward r_t for that transition.

A. Cassie Simulation Environment

We developed a simulation environment for reinforcement learning on Cassie, which is based on an open source MuJoCo simulator [37], [38]. This subsection introduces the design of the simulation environment which the reinforcement learning agent interacts with.

1) Action Space: The action $\mathbf{a}_t = q_m^d$ specifies target positions for the 10 motors on Cassie. In order to obtain a smoother motion, the target positions are first passed through a low-pass filter [23], as shown in Fig. 2, before being applied to the motors. A joint-level PD controller generates torque u for each motor on Cassie based on the filtered targets.

2) State Space: The state $\mathbf{s}_t = (\mathbf{q}^o_{t-4:t}, \mathbf{a}_{t-4:t-1})$ at time t consists of two components. The first component consists of the observable robot states $\mathbf{q}^o = [q^o, \dot{q}^o]$ at the current time step t and the past 4 time steps. The second component consists of the actions $\mathbf{a}$ from past 4 time steps. This history of past observations and actions provides the policy with more information to infer the system dynamics.

3) Goal: To train a policy to produce a desired reference motion, target frames from the reference motion are provided to the policy as input via a time-dependent goal $\mathbf{g}_t$. The user command c is used to operate the robot online and it is defined as $c = [p^d \ \dot{q}^d_\phi] = [\dot{q}^d_x \ \dot{q}^d_y \ q^d_z, \dot{q}^d_\phi]$ which includes desired gait parameters p^d and desired turning yaw velocity $\dot{q}^d_\phi$. Given a desired gait parameter p^d, a reference gait g^r is constructed by interpolating the parameterized gait library with respect to p^d as explained in Sec. II-B. The goal is then specified by $\mathbf{g}_t = (c(t), g^r(t), g^r(t+1), g^r(t+4), g^r(t+7))$, which includes 1) the current user commands, and 2) reference motor positions q^m_r and velocity $\dot{q}^r_m$ for current and sampled future time steps.

B. Reward Function

The reward function is designed to encourage the agent to satisfy the given command while reproducing the corresponding reference motion from the gait library on the dynamic robot system. The reward at each time step t is given by:

$$r_t = \omega r^o_t \tag{1}$$

$$r^o_t = [r^m_t, r^p_t, r^{\dot{p}}_t, r^r_t, r^{\dot{r}}_t, r^u_t, r^f_t]^T \tag{2}$$

$$\omega = [0.3, 0.24, 0.15, 0.13, 0.06, 0.06, 0.06] \tag{3}$$

The motor reward r^m_t encourages the policy to minimize the discrepancies between the actual motor positions $\hat{q}_m$ and the reference motion q^r_m and is formulated as:

$$r^m_t = \exp[-\rho_1 ||q^r_m - \hat{q}_m||^2_2]. \tag{4}$$

where ρ_i is a scaling factor for the i^{th} reward term. The reward terms r^p_t, $r^{\dot{p}}_t$ and $r^{\dot{r}}_t$ follow the same formulation as (4) and encourage the agent to track reference pelvis translational position, translational velocity and rotational velocity in robot local frame, respectively. The pelvis rotation reward r^r_t leads Cassie to reduce the difference between the reference rotation q^r_r and the actual one $\hat{q}_r$, and it is formulated by $r^r_t = \exp[-\rho_4 ||q^r_r \ominus \hat{q}_r||^2_2]$ where $\ominus$ denotes the geodesic distance between two rotation angles. The torque reward $r^u_t = \exp[-\rho_6 ||u||^2_2]$ encourages the robot to reduce energy consumption. Lastly, the ground reaction force reward $r^f_t = \exp[-\rho_7 ||\hat{q}_f||^2_2]$ helps to minimize the vertical contact forces $\hat{q}_f$. The weights of each reward term in ω_i are specified manually.

The desired roll and pitch velocity are always set to 0 to stabilize the pelvis, while the desired yaw velocity is specified by the user command $c(t)$. Furthermore, since no desired position terms, *e.g.*, pelvis translational and rotational positions, are explicitly given, they are computed by integrating the corresponding desired velocity.

Note that the reference motion from the gait library does not encode turning yaw information. Therefore, including non-zero desired turning yaw in the reward can encourage the agent to develop walking behaviors that are not provided by the reference motions.

C. Domain Randomization

In order to improve the robustness of the policy and bridge the gap between the simulation and the real world, the dynamics of the environment is randomized during training in simulation. The randomization regiment is designed to

TABLE II: Dynamics Properties and Sample Range.

Parameter	Range	Unit
Link Mass	[0.75,1.15] × default	kg
Link Mass Center	[0.75,1.15] × default	m
Joint Damping	[0.75,1.15] × default	Nms/rad
Ground Friction Ratio	[0.5, 3.0]	1
Motor Rotation Noise	[-0.1, 0.1]	rad
Motor Angle Velocity Noise	[-0.1, 0.1]	rad/s
Accelerometer Noise	[-0.4, 0.4]	m/s^2
Gyro Rotation Noise	[-0.1, 0.1]	rad
Gyro Angle Velocity Noise	[-0.1, 0.1]	rad/s
Communication Delay	[0, 0.03]	sec

address three major sources of uncertainty in the environment: 1) modelling error of the robot and the environment, 2) sensor noise, and 3) communication delay between the policy and the joint-level controller. These dynamics properties are parameterized as $\boldsymbol{\mu}$, whose values are varied between the ranges specified in Tab. II.

D. Learning Model

The objective of reinforcement learning is to maximize the total expected reward over trajectories $\tau \sim p_\theta(\tau)$

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=0}^{T} \gamma^t r_t \right] \tag{5}$$

where $p_\theta(\tau)$ is the distribution of trajectories $\tau = \{\mathbf{s}_0, \mathbf{a}_0, r_0, \ldots, \mathbf{s}_T, \mathbf{a}_T, r_T\}$ subject to policy $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t, \mathbf{g}_t)$, θ is the parameters of the network policy, γ is the discount factor, and T is the horizon of each episode. We use Proximal Policy Optimization (PPO) [39] to train the policy in simulation with networks with 2 hidden layers of 512 $tanh$ units for both the policy and value function.

For the policy network, the input contains observed state $\mathbf{s}_t^o$ and goal $\mathbf{g}_t$, as formulated in Sec. III-A. In $\mathbf{s}_t^o$, the observed robot state $\mathbf{q}_t^o$ is with added noise and delay introduced in Sec. III-C. The policy network uses $tanh$ as the activation function for the last layer. The output of the policy network is a 10 dimensional vector represented by a Gaussian action distribution $\mathcal{N}(\mathbf{m}_\pi(\mathbf{a}|\mathbf{s}), \mathbf{\Sigma}_\pi)$ with a learned mean $\mathbf{m}_\pi(\mathbf{a}|\mathbf{s})$ and fixed standard deviation $\mathbf{\Sigma}_\pi = 0.1I$. The action $\mathbf{a}$ (*i.e.*, desired motor positions) is sampled from this output distribution.

The value network outputs a scalar value $V(\mathbf{s}_t, \mathbf{g}_t)$ representing the expected return of the policy given state $\mathbf{s}_t$ and goal $\mathbf{g}_t$. The value network is provided access to the ground truth state $\mathbf{s}^{gt}$, Moreover, the randomized dynamic parameters $\boldsymbol{\mu}$ described in Sec. III-C are also provided as inputs to the value function.

E. Training Setup

The policy operates at 30 Hz, while the joint-level PD controller illustrated in Fig. 2 runs at 2000 Hz. The maximum number of time steps for each episode is designated to be T=2500, corresponding to approximately 83 s. In each episode, a new command $c(t) = [\dot{q}_x^d \;\; \dot{q}_y^d \;\; q_z^d, \dot{q}_\phi^d]$ is uniformly sampled every 8 s, and remains unchanged during the 8 s window. The command range is from $[-2, -0.8, 0.65, -\pi/6]^T$ to $[2, 0.8, 1, \pi/6]^T$. The first command in each episode is always set to a random walking forward velocity with 0 yaw velocity at a normal height above 0.9 m. Note that the range of training commands is larger compared to the gait parameter provided in Tab. I. In this way, the agent is able to learn to follow a command that is out-of-range of gait parameters and thus learns behaviors beyond what the gait library can provide. An episode ends when the maximum number of time steps is reached, or early termination conditions have been triggered. Early termination is triggered if the height of the pelvis drops below 0.55 m, and if the tarsus joints $q_5^{L/R}$ hit the ground.

Dynamics randomization presented in Sec. III-C is introduced gradually over the course of training through a curriculum. The curriculum helps to prevent the policy from adopting excessively conservative sub-optimal behaviors. For example, if training starts with the full range of randomizations detailed in Table II, the policy is prone to adopting strategies that prevent the robot from falling by simply standing in-place. In a highly dynamic environment, standing can be more stable than walking, and is therefore easier to learn, but is nonetheless sub-optimal. Therefore, over the course of the first 2000 training iterations, the upper and lower boundaries of the randomized dynamics parameters are linearly annealed from fixed default values to the maximum ranges specified in Tab. II.

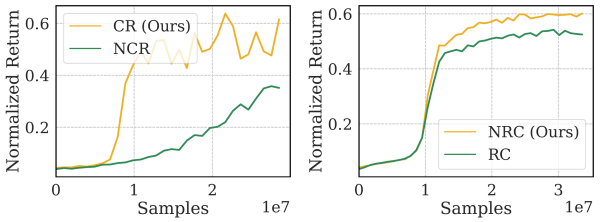
IV. Experiments

The walking policy is trained with a MuJoCo simulation of the Cassie robot [37]. The performance of the learned policy is evaluated in three domains: MuJoCo, MATLAB SimMechanics, and on Cassie. Performance is first evaluated in the MuJoCo simulator, which is the domain used for training. Later, SimMechanics provides a safe and high-fidelity simulated environment that closely replicates the physical system to extensively test the learned policy. However, the high-fidelity simulation is slower than real-time by an order of magnitude, so it is primarily used for testing. Finally, the policy is deployed and validated on the real Cassie robot.

A. Learning Performance

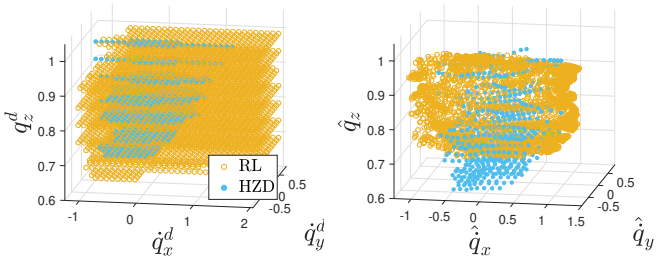
To evaluate the effects of the randomization curriculum, we compare the performance of policies trained with and without the curriculum. Fig. 3a compares learning curves for the different policies. The policy trained with the randomization curriculum (CR) starts training with a small amount of randomization, which is then gradually increased over the course of training. The policy trained without the curriculum (NCR) starts training with the full range of randomization. The large amount of randomization at the start of training leads the policy to adopt an excessively conservative and sub-optimal behavior. The policy trained with the curriculum exhibits substantially faster learning progress, while also achieving a higher return.

The effects of residual control are evaluated by comparing the performance of our policy that uses non-residual control (NRC), with a policy that uses residual control (RC) [4], [5]. Learning curves comparing the different policies are available in Fig. 3b. In the absence of external perturbations,



(a) (b)

Fig. 3: Comparison between (a) proposed Curriculum (CR) and Non-Curriculum (NCR) methods and (b) proposed Non-Residual Control (NRC) and Residual Control (RC) used in previous work [5]. Our proposed method shows best overall training performance in terms of learning speed and converged rewards. The corresponding total samples for the curriculum is $6.6e7$, while the NCR model has full range of dynamics randomization from the start of training.



(a) Feasible Command Set (b) Safe Set

Fig. 4: Comparison of proposed commands (Feasible Command Set) and achieved commands (Safe Set) between HZD-based controller [17] and proposed RL-based controller. Our RL-based controller can handle more tracking commands than the HZD-based baseline and thus results in a larger feasible command set. The safe set of RL-based policy is also larger in the sagittal walking velocity $\hat{q}_x$ and walking height $\hat{q}_z$ direction. The tracking performance of the RL-based policy also shows advantages as the shapes of feasible command set and safe set are closer.

the performance of the two policies is similar, with the non-residual policy performing marginally better than the residual policy. However, as we show in the following experiments, our non-residual policy with dynamics randomization is more robust than the residual policy, which may be due to the non-residual policy's greater flexibility to deviate from the behaviors prescribed by the reference motion in order to recover from perturbations.

B. Robustness Analysis in High-Fidelity Simulation

A *Feasible command set* is a set of input gait parameters p^d that will not cause a controller to fail. A *safe set* is defined as a set of gait parameters that a controller actually achieves on the robot while maintaining a stable walking gait. During each iteration, a gait parameter p^d is provided to the controller as a command in MATLAB SimMechanics, if the controller succeeds in maintaining a stable gait for 15 seconds, then p^d will be added to the feasible command set and the actual achieved gait parameter $\hat{p}$ will be added to the safe set. Through extensive tests of a controller, the resulting feasible command set and safe set provide informative metrics to evaluate the control performance and robustness of the controller when deployed on the robot. Typically, a walking controller with a larger feasible command set can handle more scenarios, and a controller with a larger safe set can achieve more dynamic motions. Moreover, a controller with better tracking performance can result in a similar shape between the feasible command set and safe set, as the difference between these two sets indicates tracking errors between p^d and $\hat{p}$.

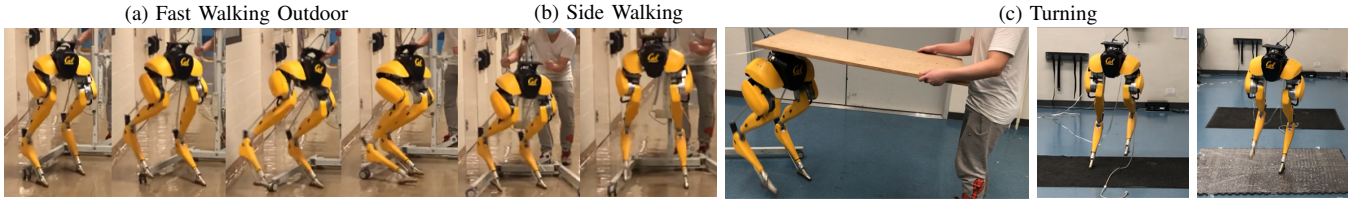
We compare the feasible command sets and safe sets between the learned policy and prior HZD-based variable walking height controller developed in [17] and based on [16]. The procedures for generating the command sets and safe sets of these two controllers are identical, and the testing range for p^d is set to be between $[-1.1,\ -0.6,\ 0.65]^T$ and $[2,\ 0.6,\ 1.0]^T$, with a resolution of $[0.1,\ 0.1,\ 0.05]^T$. The resulting feasible command sets and safe sets are shown in Fig. 4. As shown in Fig. 4a, the proposed RL-based controller is able to cover almost the entire testing range, while the HZD-based controller can only handle a smaller bowl-shape region. Quantitatively, the feasible command set of the RL-based walking controller is more than 4 times larger than HZD-based controller. Moreover, as illustrated in Fig. 4b, the RL-based walking controller covers a broader safe set than the HZD-based controller. In practice, this means that the RL-based controller can achieve faster forward and backward walking (from -1.2 m/s to 1.2 m/s) than HZD-based one (below 1 m/s). Although Fig. 4b shows that the HZD-based controller can achieve walking gaits with lower heights (0.6 m) than the RL-based one (0.65 m), the tracking error of the HZD-based controller is not negligible as its minimum feasible walking height command can only reach 0.7 m while RL-based one can go to 0.65 m. Therefore, the RL-based controller exhibits better performance on tracking commands. Moreover, by inspecting the relationship between Fig. 4a and Fig. 4b, we find that the RL-based controller can handle the commands that are outside of the given gait library in Tab. I, *e.g.*, 2 m/s in the sagittal direction. The resulting actual velocity $\hat{q}_x$ is around 1.2 m/s, which is also outside of the range seen in the gait library. With the standard HZD-based controller, the robot only approaches 1 m/s when it is being given a 2 m/s command, due to a large tracking error. This shows that the RL-based controller is not strictly tracking the commands, and instead finds a more optimal gait that is close to the commands while maintaining stability.

C. Robustness in the Real World

The deployed policy on Cassie in the real world can reliably control the robot to perform various behaviors, such as changing walking heights in Fig. 1a,1b, fast walking in Fig. 5a, walking sideways in Fig. 5b, turning around in Fig. 5c. Moreover, the policy also shows robustness to the changes of the robot itself and the environment.

1) Modeling Error: During the experiments in this paper, a malfunction caused two motors on the Cassie to not work properly. Specifically, the right rotation q_2^R and right knee q_4^R motors were partially damaged, making them unable to produce as much torque as the corresponding motors on the left side or in the simulation. Following this mal-

(a) Fast Walking Outdoor (b) Side Walking (c) Turning



(d) Recover from Foot Sliding (e) Unknown Load (f) Anti-Slip (g) Slippery

Fig. 5: Experiment Results. The proposed learned walking policy extensively on Cassie in real world in different scenarios. In the experiments, the policy enables the Cassie to perform various agile behaviors such as fast forward and backward walking, sideways walking, changing walking height, and turning around. Moreover, empowered by the proposed policy, the robot is able to recover from random perturbation and also able to adapt to change different ground frictions and unknown load.

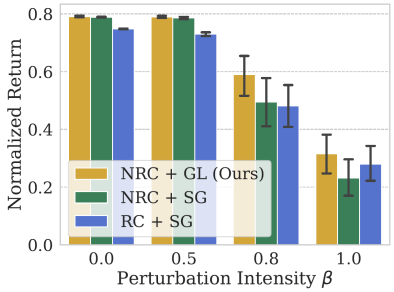


Fig. 6: Comparison of robustness to perturbation among 3 different methods in MuJoCo simulation. Non-Residual Controlled Gait Library (NRC+GL) is trained with the gait library; Non-Residual Controlled Single Gait (NRC+SG) and Residual Controlled Single Gait (RC+SG) are trained with only one single gait, which is also the test motion. A 6 DoF force is randomly applied on the pelvis with probability $0.15\beta\%$. The *Normalized Return* is computed by the mean reward of 32 roll-outs for each model for each β.

function, model-based walking controllers, such as the factory default controller and the HZD-based variable walking controller [17], were no longer able to reliably produce a walking gait. The baseline HZD-based controller was no longer able to recover to a normal height after a reduction in walking height, since the right leg was weaker than the left leg. However, by training with dynamics randomization (Sec. III-C), especially the damping ratio of each joint, the proposed learned walking controller could control the robot even with partially damaged motors. Indeed, this policy was able to successfully control the robot the very first time it was deployed, without additional tuning.

2) Perturbation: To show that our approach is more robust, three quantitative experiments are done in the MuJoCo simulator: 1) a non-residual policy trained with the gait library, 2) a non-residual policy trained using a single reference motion from the gait library, and 3) a residual policy trained with the same single gait as the previous work [5]. All policies are trained without domain randomization. During the evaluation, the pelvis is perturbed randomly by a 6 DoF force with a probability of $0.15\%\beta$ at each time step lasting for a random time span sampling from $[0, 0.8\beta]$ s, where $\beta \in [0, 1]$ stands for perturbation intensity. The achieved return of each model is illustrated in Fig. 6. When larger perturbations like $\beta \in \{0.8, 1\}$ are applied, the model trained by the proposed method shows significant advantages over other models.

To further demonstrate the robustness in the real world qualitatively, we randomly push Cassie with a rod in different directions, *e.g.* from the front of the pelvis in Fig. 1c, from the back in Fig. 1d, and from left and right of the pelvis. The feet of Cassie are also perturbed during walking, including stepping on the gantry in Fig. 5d. In addition an unknown load is applied in Fig. 5e and changes in ground friction in Fig. 5f,5g. The proposed learned policy shows improved robustness over previous work across all scenarios.

V. Conclusion and Future Works

To our knowledge, this paper is the first to develop a diverse and robust bipedal locomotion policy that can walk, turn and squat using parameterized reinforcement learning. In this work, a model-free reinforcement learning method is proposed to train a policy that is able to control Cassie to track given walking velocities, walking heights, and turning yaw velocities, by imitating reference motions decoded from a HZD-based gait library. In contrast to prior work on reinforcement learning for bipedal locomotion with Cassie, our method does not utilize a residual control term, providing improved flexibility, and instead uses the HZD-based gait library to provide references for diverse training. This results in better performance and robustness, as well as more sophisticated recoveries. The proposed learning method shows benefits over a baseline model-based walking controller, producing a larger feasible command set, a larger safe set, and better tracking performance. In the real world experiments, the policy also demonstrates considerable robustness, effectively controlling Cassie, even with malfunctioning motors, to walk over floors with different friction and rejecting perturbations. An exciting future direction is to explore how more dynamic and agile behaviors can be learned for Cassie, building on the approach presented in this work.

REFERENCES

[1] M. Vukobratovic and B. Borovac, "Zero-moment point—thirty five years of its life," *International journal of humanoid robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[2] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The international journal of robotics research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[3] J. W. Grizzle, C. Chevallereau, A. Ames, and R. Sinnet, "3d bipedal robotic walking: Models, feedback control, and open problems," in *IFAC Symposium on Nonlinear Control Systems*, 2010.

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

[5] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. Panne, "Learning locomotion skills for cassie: Iterative design and sim-to-real," in *Conference on Robot Learning*, 2020, pp. 317–329.

[6] 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," in *International Conference on Robotics and Automation*, 2019, pp. 6023–6029.

[7] S. Kuindersma, F. Permenter, and R. Tedrake, "An efficiently solvable quadratic program for stabilizing dynamic locomotion," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2589–2594.

[8] J. Pratt, T. Koolen, T. De Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid," *The international journal of robotics research*, vol. 31, no. 10, pp. 1117–1133, 2012.

[9] X. Xiong and A. D. Ames, "Coupling reduced order models via feedback control for 3d underactuated bipedal robotic walking," in *IEEE-RAS International Conference on Humanoid Robots*, 2018.

[10] T. Koolen, J. Smith, G. Thomas, S. Bertrand, J. Carff, N. Mertins, D. Stephen, P. Abeles, J. Englsberger, S. Mccrory *et al.*, "Summary of team ihmc's virtual robotics challenge entry," in *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE, 2013, pp. 307–314.

[11] S. Feng, X. Xinjilefu, W. Huang, and C. G. Atkeson, "3d walking based on online optimization," in *2013 13th IEEE-RAS International Conference on Humanoid Robots*, 2013, pp. 21–27.

[12] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE-RAS International Conference on Humanoid Robots*, 2014, pp. 295–302.

[13] A. Hereid, O. Harib, R. Hartley, Y. Gong, and J. W. Grizzle, "Rapid trajectory optimization using c-frost with illustration on a cassie-series dynamic walking biped," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 4722–4729.

[14] X. Da, O. Harib, R. Hartley, B. Griffin, and J. W. Grizzle, "From 2d design of underactuated bipedal gaits to 3d implementation: Walking with speed tracking," *IEEE Access*, vol. 4, pp. 3469–3478, 2016.

[15] Q. Nguyen, X. Da, W. Martin, H. Geyer, J. W. Grizzle, and Sreenath, "Dynamic walking on randomly-varying discrete terrain with one-step preview," in *Robotics: Science and Systems*, 2017.

[16] Y. Gong, R. Hartley, X. Da, A. Hereid, O. Harib, J.-K. Huang, and J. Grizzle, "Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway," in *American Control Conference*, 2019, pp. 4559–4566.

[17] Z. Li, C. Cummings, and K. Sreenath, "Animated cassie: A dynamic relatable robotic character," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[18] S. Coros, A. Karpathy, B. Jones, L. Reveret, and M. Van De Panne, "Locomotion skills for simulated quadrupeds," *ACM Transactions on Graphics (TOG)*, vol. 30, no. 4, pp. 1–12, 2011.

[19] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–13, 2017.

[20] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, pp. 1–14, 2018.

[21] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation, 2004.*, vol. 3. IEEE, 2004, pp. 2619–2624.

[22] 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, 2019.

[23] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals," in *Robotics: Science and Systems*, 07 2020.

[24] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *IEEE-RAS International Conference on Humanoid Robots*, 2007, pp. 392–397.

[25] T. Li, N. Lambert, R. Calandra, F. Meier, and A. Rai, "Learning generalizable locomotion skills with hierarchical reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 413–419.

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

[27] J. Morimoto and C. G. Atkeson, "Learning biped locomotion," *IEEE Robotics & Automation Magazine*, vol. 14, no. 2, pp. 41–51, 2007.

[28] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3503–3510.

[29] G. A. Castillo, B. Weng, T. C. Stewart, W. Zhang, and A. Hereid, "Velocity regulation of 3d bipedal walking robots with uncertain dynamics through adaptive neural network controller," *arXiv preprint arXiv:2008.00376*, 2020.

[30] G. A. Castillo, B. Weng, W. Zhang, and A. Hereid, "Hybrid zero dynamics inspired feedback control policy design for 3d bipedal locomotion using reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 8746–8752.

[31] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[32] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 23–30.

[33] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE international conference on robotics and automation*, 2018, pp. 1–8.

[34] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2950–2957, 2020.

[35] J. Siekmann, S. Valluri, J. Dao, L. Bermillo, H. Duan, A. Fern, and J. Hurst, "Learning memory-based control for human-scale bipedal locomotion," *arXiv preprint arXiv:2006.02402*, 2020.

[36] G. C. Haynes and A. A. Rizzi, "Gaits and gait transitions for legged robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2006, pp. 1117–1122.

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

[38] Agility Robotics, "cassie-mujoco-sim. (2018) [online]." [Online]. Available: https://github.com/osudrl/cassie-mujoco-sim

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