GLiDE: Generalizable Quadrupedal Locomotion in Diverse Environments with a Centroidal Model

Zhaoming Xie[1,2], Xingye Da[2], Buck Babich[2], Animesh Garg[2,3], Michiel van de Panne[1]
[1]University of British Columbia, [2]Nvidia, [3]University of Toronto & Vector Institute

***Abstract*—Model-free reinforcement learning (RL) for legged locomotion commonly relies on a physics simulator that can accurately predict the behaviors of every degree of freedom of the robot. In contrast, approximate reduced-order models are often sufficient for many model-based control strategies. In this work we explore how RL can be effectively used with a centroidal model to generate robust control policies for quadrupedal locomotion. Advantages over RL with a full-order model include a simple reward structure, reduced computational costs, and robust sim-to-real transfer. We further show the potential of the method by demonstrating stepping-stone locomotion, two-legged in-place balance, balance beam locomotion, and sim-to-real transfer without further adaptations. Additional Results: https://www.pair.toronto.edu/glide-quadruped/.**

I. Introduction

Tremendous progress has been made recently in the field of legged locomotion, achieved using both model predictive control (MPC) and reinforcement learning (RL) methods. MPC methods leverage modern optimization techniques and known models of the physics, possibly simplified, to synthesize responsive control at run time. However, they can be prone to local minima, can require substantial manual tuning, and are difficult to generalize to complex terrains and rich perceptual streams. Moreover, the real-time MPC usually uses a linear model to reduce the computation time, which is hard to represent the legged system's nonlinear and hybrid nature. Alternatively, model-free methods such as reinforcement learning (RL) utilize Monte-Carlo sampling strategies and can learn control policies for general tasks. This comes at the expense of extensive offline physics simulations required during training, careful system modeling, and detailed reward design to produce results that are feasible for physical robots.

In this paper, we seek to realize some of the key benefits of both approaches. Instead of relying on an accurate simulation of the robot model, we use a strongly-abstracted centroidal model. We model the robot as a single rigid body with massless legs that is controlled via the ground reaction forces (GRF) applied at the legs that are in contact with the environment, as shown in Fig. 1. We further assume a specified gait pattern and a foot-placement function. Taken together, this allows for a simple task reward specification, in contrast to the more complex reward structures commonly required for full-model RL. Additional constraints such as no-slip constraints and leg lengths are enforced via quadratic programming (QP) and foot placement strategies. With the above in place, we then synthesize control policies for this simplified model

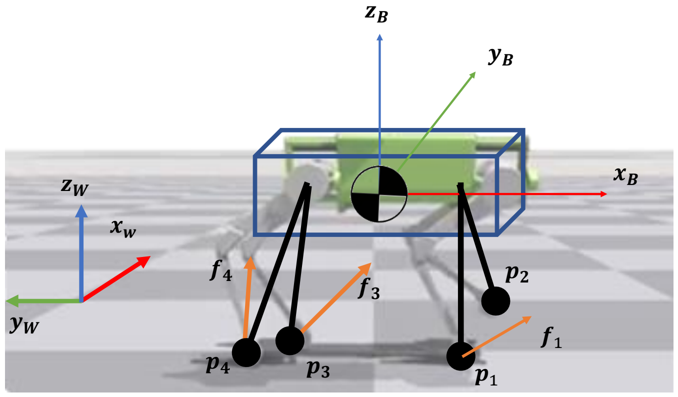


Fig. 1: Learning to generate a whole-body locomotion controller with reinforcement learning is both inefficient and brittle. A key insight of this paper is to combine the strengths of reduced-order modeling with learning through RL in Centroidal Model Space. The centroidal model consists of a single rigid body with virtual legs attached as illustrated above. Control is realized by generating ground reaction forces at foot locations in contact with the ground.

via reinforcement learning. The resulting policy actions are realized on the full robot model by converting the GRFs to joint torques using the Jacobian transpose for the stance legs and a simple trajectory-tracking approach for the swing legs. A system overview is given in Fig. 2. The resulting control policies are validated on simulations of the Laikago and A1[1] quadruped robots as well as on a physical A1 robot.

Our core contributions are as follows:

- We introduce a framework for learning control policies suited for centroidal dynamics models, enabling the anticipatory behavior required for quadrupedal locomotion and balance tasks, without the complexities of working with the full model. This allows for simple reward design, enables efficient simulation during training, and yields flexible and robust motion control.
- We demonstrate the effectiveness of this framework for multiple gaits, stepping stone scenarios, balance beam locomotion, and two-legged in-place balancing. We further show successful transfer to a physical robot.

II. Related Work

Our work combines techniques used in quadrupedal locomotion in MPC and learning literature to efficiently solve challenging tasks. A comparison with some key papers is

[1]Laikago and A1 are quadrupedal robots made by Unitree Robotics.

TABLE I: We compare with recent work on quadrupedal locomotion. Model-based control such as MPC often uses a centroidal model for efficient control synthesis but frequently fails to generalize to challenging scenarios. RL makes use of accurate full-body simulations and can handle challenging scenarios at the expense of significant computational cost and complicated reward structures. We use the centroidal model for reinforcement learning, which allows us to efficiently train policies for challenging tasks with a simple reward function.

	Model Used	**Reward/Cost Function**	**Demonstrated Tasks**	**Training Time (Flat Terrain)**
Centroidal MPC [1, 2]	Linearized Centroidal	Linear Quadratic Regulator (✓)	Flat, Back Flip	n/a
Centroidal PD [3, 4, 5]	Centroidal	n/a	Flat	n/a
Laikago RL [6, 7]	Full Physics	5 terms (✗)	Flat	6-8 hours
Anymal RL [8, 9, 10]	Full Physics	7-11 terms (✗)	Flat, Get Up, Rough Terrain	3-4 hours
DeepGait [11]	Centroidal + Full Physics	7 terms (✗)	Flat, Stepping Stone, Stairs	58 hours
GLiDE (RL) (ours)	Centroidal	2 terms (✓)	Flat, Two-Legged Balancing Stepping Stone, Balance Beam	1-2 hours

provided in TABLE I. In this section we review more related work in this area.

A. Reduced-Order Models for Legged Locomotion

Due to the complexity of legged robots, model-based control approaches often make use of reduced-order models for control synthesis. An inverted pendulum and variations thereof are commonly used as abstract models of bipedal robots to synthesize control policies, e.g., [12, 13, 14, 15, 16]. A centroidal model, consisting of a single rigid body driven by external forces, is often used to generate control policies for quadrupeds, e.g., [1, 2, 5]. To further simplify control synthesis, reference motions based on these models are generated offline, and feedback is then realized around these via PD control or model-predictive control (MPC). Direct nonlinear optimization can also be used for MPC at the cost of a slower control frequency [17] or additional heuristic objectives to guide the optimization [18] out of local minima. In this paper, we employ the centroidal dynamics model for reinforcement learning. The resulting policy directly optimizes for long-horizon nonlinear problems without the aid of a reference trajectory or heuristic cost.

B. Deep Reinforcement Learning for Quadrupedal Robots

Deep RL has become a viable approach for synthesizing control policies for quadrupedal robots. It is often realized via a comprehensive simulation of the robot, e.g., [9, 19, 20]. Rewards designed based on reference trajectories [6, 7] or carefully tuned reward terms [8, 10, 21, 22] are often necessary to regularize undesirable behaviors as to be feasible for a physical robot. The computation cost to train a policy often requires millions to billions of transition tuples of the full physics simulation. In this paper, we do not rely on a multibody dynamics simulation to train our control policy. Instead, we simulate a low-cost centroidal dynamics model, which can be easily parallelized without resorting to special-purpose GPU physics simulators, e.g., [23, 24]. When used in combination with a simple foot placement strategy, it requires significantly less computation for scenarios that requires complex contact modeling, such as stepping stone and balance beam traversal, as compared to recent work, e.g., [11, 25].

Our work has some parallels with model-based RL, as we do not rely on a full physics simulator or a physical robot to perform Monte Carlo rollouts. Recent work using model-based RL for legged robot control still needs to collect simulation data or physical robot data to learn either a transition model that contains the full state information [26] or center of mass (COM) information [27]. We rely instead on the simple centroidal dynamics to perform Monte Carlo rollouts, which does not require data from a full model or a physical robot. Our simulation is easily adapted to robots with different physical parameters or morphology by changing the compact set of parameters that define the centroidal model, such as the mass, inertia, and leg length.

C. Combination of Model-Based Control and Learning

Model-based control comes with the benefit of explainability but often at the expense of significant online computation, manual tuning, and feature engineering. Alternatively, learned RL policies enable fast online computation and can in principal cope with complex terrains and rich perceptual observations but suffer from expensive offline computation and a control policy that is highly specific to the given robot morphology.

A combination of model-based control and RL offers best of both approaches. A combination of RL in task spaces in combination with a model-based controller has yielded generalization in learning for manipulation skills [28, 29]. A model-based controller can be used to collect data offline and then efficiently queried online using supervised learning, e.g., [30, 31, 32]. Heuristics or dynamics can be learned to enable faster optimization or planning, e.g., [18, 33]. Hierarchical control structures have been proposed to allow model-based control and model-free RL policies to operate at different time scales to leverage their respective advantages, e.g., [4, 34, 35, 36].

Our work also aims to combine RL and model-based control. A high-level learned policy operates in the space of centroidal dynamics, producing desired linear and angular accelerations as output. The low-level control policy then uses model-based methods to realize these high-level commands, aided by a quadratic program that transcribes the desired accelerations into appropriate ground reaction forces.

III. GLiDE: RL with Centroidal Model

In this section, we describe GLiDE for learning policies using a centroidal dynamics model with virtual legs and the related approach for realizing the resulting policy on the full robot model. We describe the centroidal dynamics simulation, the control of the centroidal dynamics by transforming the desired body acceleration to ground reaction forces via a quadratic program (QP), and the use of a Raibert-style heuristic for foot placement. Further, we detail the procedure for training RL policies using GLiDE, and policy transfer to full-order model using model-based control. The main algorithm for simulating the centroidal model is outlined in Algorithm 1, and the control structure is outlined in Fig. 2.

A. Centroidal Dynamics Model With Virtual Legs

Our centroidal dynamics model consists of a single rigid body with four massless legs attached; see Fig. 1. The rigid body has mass m and inertia $I \in \mathbb{R}^{3\times3}$. Each leg has a phase variable ϕ that advances linearly in time. The state of the rigid body $s = [p, \dot{p}, R, \omega]$ consists of the linear position and velocity $p, \dot{p} \in \mathbb{R}^3$, orientation $R \in SO(3)$, and angular velocity $\omega \in \mathbb{R}^3$. Following [2], we represent R with a rotation matrix.

We follow the derivation from [2] to simulate the centroidal dynamics. For completeness, we describe our notation and key equations used for simulation. For each leg i, ground reaction forces $f_i \in \mathbb{R}^3$ can be generated to drive the evolution of the state of the rigid body. When walking on flat ground, if leg i is in swing phase, $f_i = 0$; otherwise f_i needs to obey the friction cone constraints in order to prescribe feasible forces. Given the foot position $p_i \in \mathbb{R}^3$ and f_i for each leg i, the net force and torque $f_{\text{net}}, \tau_{\text{net}} \in \mathbb{R}^3$ can be calculated as:

$$\begin{aligned} f_{\text{net}} &= \sum_{i=1}^{4} f_i - mg \\ \tau_{\text{net}} &= \sum_{i=1}^{4} \hat{r}_i f_i \end{aligned} \tag{1}$$

where g is the gravitational constant, $r_i = p_i - p$, and $\hat{\cdot} : \mathbb{R}^3 \to \mathfrak{so}(3)$ defines a mapping from a vector to its skew-symmetric matrix form.

The Euler integration update of the rigid body state given time step Δt can then be written as

$$\begin{aligned} p &= p + \dot{p}\Delta t \\ \dot{p} &= \dot{p} + \frac{1}{m} f_{\text{net}} \Delta t \\ R &= R \exp(\widehat{\omega \Delta t}) \\ \omega &= \omega + I^{-1}(R^T \tau_{\text{net}} - \hat{\omega} I \omega)\Delta t \end{aligned} \tag{2}$$

Algorithm 1 Centroidal Dynamics Simulation

```
 1: Initialize centroidal state $s = [p, \dot{p}, R, \omega]$, foot position
    $p_{\text{foot}} = [p_1, p_2, p_3, p_4]$, foot phases $\phi_{\text{foot}} = [\phi_1, \phi_2, \phi_3, \phi_4]$.
 2: while Simulate do
 3:     Compute desire acceleration from a policy $a_d = [\ddot{p}_d, \dot{\omega}_d]$.
 4:     Generate ground reaction forces by solving the QP
        problem: $[f_1, f_2, f_3, f_4] = \arg\min QP$ ▷ Equation 3.
 5:     Compute the net force and torque $f_{\text{net}}, \tau_{\text{net}}$ ▷ Equation 1.
 6:     Compute the resulting acceleration $a = [\ddot{p}, \dot{\omega}]$.
 7:     Update centroidal state $s = [p, \dot{p}, R, \omega]$ ▷ Equation 2.
 8:     Advance foot phases $\phi_{\text{foot}} = [\phi_1, \phi_2, \phi_3, \phi_4]$.
 9:     if foot $i$ needs to update position then
10:         Update $p_i$ ▷ Equation 4.
11:     end if
12: end while
```

The exponential map $\exp : \mathfrak{so}(3) \to SO(3)$ ensures that R stays on the $SO(3)$ manifold. We set $\Delta t = 0.01$ s for our experiments.

The phase variable ϕ_i for each foot i is set to be an integer that increments by 1 at each simulation time step. We define an integer cycle time, T, and an integer swing time, T_{swing}. Foot i is in swing phase if $(\phi_i \mod T) \leq T_{\text{swing}}$ and in stance phase otherwise.

B. Quadratic Programming Based Control

While the centroidal model is driven by the ground reaction forces, it is often unintuitive to directly provide the set of ground reaction forces with model-based control design. Furthermore, in the reinforcement learning setting, it is difficult to guarantee inequality constraints such as a friction cone if we choose the ground reaction forces as the control action. Instead, we choose the control action to be the desired acceleration $a_d = [\ddot{p}_d, \dot{\omega}_d]$. We then solve a QP to transcribe this into a set of ground reaction forces:

$$\begin{aligned} \min_f \; & \|Mf - a_d\|_Q^2 + \lambda \|f\|^2 \\ \text{subject to } & f_{z,i} \geq f_{z,\min} \text{ if foot i in stance phase} \\ & f_{z,i} = 0 \qquad \text{if foot i in swing phase} \\ & -\mu f_z \leq f_x \leq \mu f_z \\ & -\mu f_z \leq f_y \leq \mu f_z \end{aligned} \tag{3}$$

where $M \in \mathbb{R}^{6\times12}$ is the inverse inertia matrix, which is a function of foot position p_{foot} and the centroidal state s, $Q \succ 0$ is for weight adjustment of the cost terms, μ is the friction coefficient of the terrain and $\lambda > 0$ is to regularize the ground reaction forces used. The resulting ground reaction forces are then used to simulate the centroidal dynamics forward.

C. Foot Placement Strategy

We use a simple Raibert-style heuristic [37] for foot placement. Specifically, in the centroidal model, for each foot i, if $(\phi_i \mod T) = 0$, i.e., at the instant where it switches from

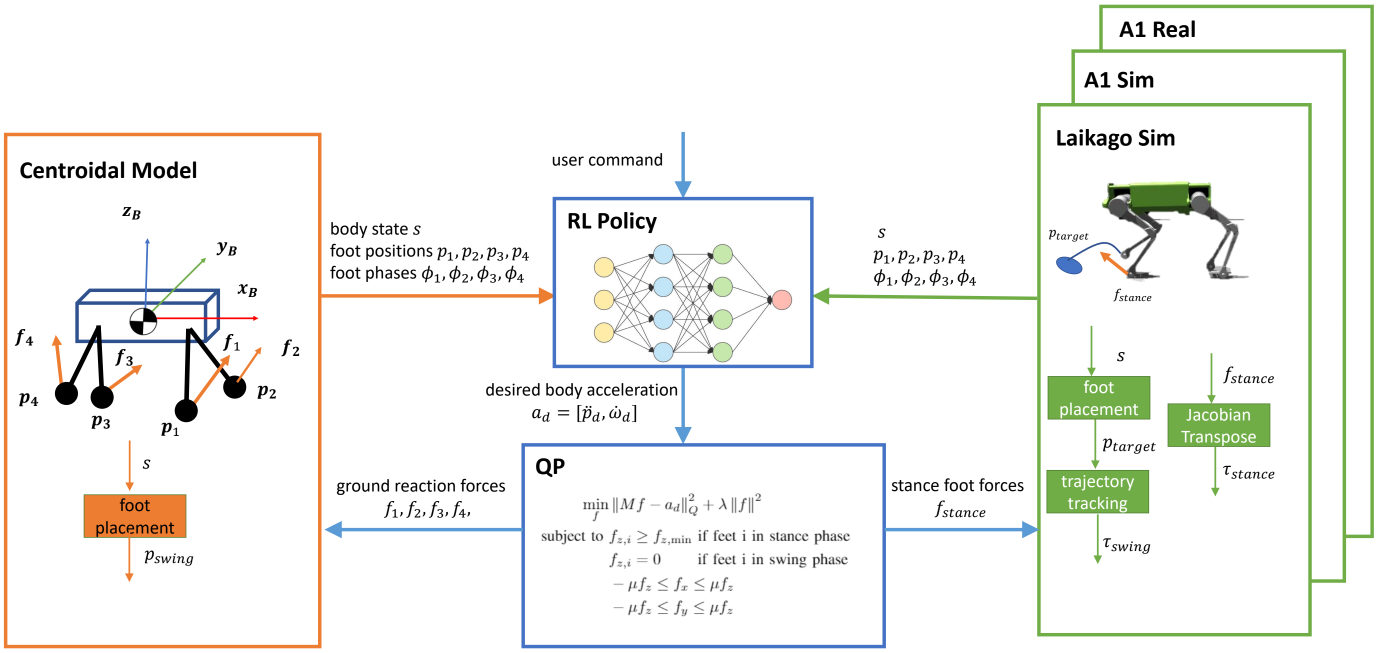


Fig. 2: Overview of our system. A control policy is trained with reinforcement learning on the centroidal dynamics of the robot. The policy generates desired body acceleration commands, which are transformed to feasible ground reaction forces using a QP. The resulting policy is then realized on the full-order model using the Jacobian transpose for stance leg control and trajectory tracking for swing leg control. All components are executed at 100 Hz.

stance foot to swing foot, the foot position relative to the body on the xy-plane, $r_{i,xy}$, will be updated based on the following strategy:

$$\begin{aligned} r_{i,x} &= r_{\text{ref},i,x} + k_{foot,x}\dot{p}_x \\ r_{i,y} &= r_{\text{ref},i,y} + k_{foot,y}\dot{p}_y, \end{aligned} \tag{4}$$

where $r_{\text{ref},i}$ are the default foot placements, typically obtained from the neutral standing pose of the robot, and k_{foot} are weights that adjust the foot placement based on the velocity of the body $\dot{p}_{xy}$. In addition, a leg-length constraint is applied such that when $\|p_{\text{foot}} - p\| \geq l_{\max}$ for some maximum leg length $l_{\max}$, the point closest to p_{foot} that satisfies the leg length constraint will be used instead.

Since the ground reaction force for the swing foot $f_i = 0$, the swing foot is assumed to not affect the dynamics of the centroidal model until it becomes the stance foot again. Importantly, however, a control policy based on the centroidal model can observe the updated position and make control decisions in anticipation of the new swing foot position.

D. Reinforcement Learning with Centroidal Model

We apply model-free RL on the centroidal model to synthesize the high-level control policy. We use an actor-critic algorithm optimized with Proximal Policy Optimization (PPO) [38]. The input to the policy includes the state of the rigid body excluding the x, y displacement, $p_z, \dot{p}, R, \omega$; the foot position relative to the center of mass position, $r_i = p_i - p$; and the foot phases ϕ_i for each foot. Reward design is entirely based on the rigid body state without concern for many of the issues that lead to more complex reward structures for RL solutions based on the full model.

Since we have the Euler integration result in analytic form, the centroidal dynamics simulation can be readily implemented on a GPU to allow thousands of parallel simulations for on-policy data collection. A bottleneck for simulation with a control policy in our setting is that we need to solve a small QP for each simulation step. We currently find no off-the-shelf QP solver that can reliably solve thousands of small QPs in parallel. We therefore implement a custom QP solver specific to our needs. The solver is implemented with PyTorch [39] for easy prototyping and is based on the interior point method [40]. The simulations are then run concurrently on a single GPU. In our experiments, we run 1600 simulations in parallel, with each environment step taking around 0.25 s on average, including reward computation and policy query. This allows for a data collection rate of 6-7k environment steps per second. The QP solves consume about 80 percent of the simulation time, thus in principle we could achieve a significant additional speedup with further improvements to the QP solver.

The benefit becomes more obvious in the presence of irregular terrain such as a field of stepping stones separated by crevices. Instead of simulating complex contact dynamics between a height field and the robot links, as in related work [10, 11], we need only select the appropriate foothold at negligible computation cost.

E. Realization on the Full Model

Since training the policy does not rely on any knowledge of how the low-level controller will be implemented on the full-order model, in principle any controller, either learned or model-based, that can generate joint commands to realize the ground reaction force required, as well as the given foot placement strategy, is applicable. Recent work uses another

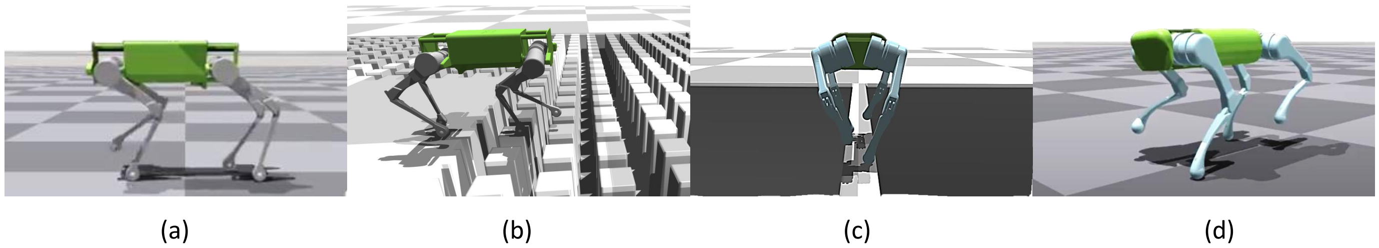


Fig. 3: We learn policies for multiple locomotion tasks, for Laikago and A1. These include: (a) locomotion on flat terrain; (b) locomotion across stepping stones; (c) locomotion across a balance beam; and (d) two-legged balancing.

learned policy to track the behavior of some reduced-order models, e.g., [13, 11], but the resulting control will only be applicable to the specific robot it is trained on. We choose to use a model-based control approach due to its data efficiency, robustness, and generality across robot morphology.

The full model control needs to realize stance foot control and swing foot control. For stance foot control, we transform the ground reaction forces into joint torques using the Jacobian transpose:

$$\tau_{\text{swing}} = J^T f,$$

where J is the stance feet position Jacobian matrix with respect to motor states.

We realize the foot placement strategy for the swing foot using a PD controller in the cartesian space of the swing foot. If $(\phi_i \mod T) = 0$, we set the swing foot target to be the updated position p_i as in the centroidal model; a spline curve parameterized by time is then constructed to connect the current foot position and p_i, with some maximum foot height clearance. The curve is then tracked via a PD controller. If early foot contact is detected, the torques for the leg are set to zero until the planned stance phase arrives. For the case of late contact, there is no special treatment; a desired ground reaction force will naturally result in leg extension.

IV. Results

To demonstrate the generality of our approach, we apply our algorithm to learn control policies for Laikago and A1, quadrupedal robots made by Unitree Robotics. The mass and inertial of the centroidal model as well as default foot placement locations and foot length constraints are adapted from the robot parameters provided by the manufacturer.

As noted previously, our method benefits from an exceptionally simple reward structure. We use the following reward specification across all the tasks presented:

$$r = 0.5 r_p + 0.5 r_o$$

where r_p regularizes the linear velocity and body height, and r_o regularizes the orientation of the body. More specifically:

$$\begin{aligned} r_p &= \exp(-10(\dot{p} - \dot{p}_{x,d})^2 - 50\dot{p}_y^2 - 50(p_z - p_{z,d})^2) \\ r_o &= \exp(-10\left\|\Theta\right\|^2), \end{aligned}$$

where $p_{z,d}$ is the nominal height of the robot based on robot parameters, and Θ is the Euler angle representation of the robot orientation. We use $p_{z,d} = 0.43\,\text{m}$ for Laikago and $p_{z,d} = 0.35\,\text{m}$ for A1. Aside from locomotion on flat terrain, we also learn control policies for challenging tasks such as stepping stone traversal, navigating across a narrow balance beam, and two-legged balancing. Example tasks are shown in Fig 3. We then test some of the policies on a physical A1 robot to demonstrate robust sim-to-real performance. Please refer to the supplementary video for example motions.

A. Trotting and Walking on Flat Terrain

We first demonstrate our system for the generation of locomotion on flat terrain. A trotting gait can be realized by synchronizing the leg phases of diagonal legs. We set the cycle time $T = 60$ and swing time $T_{\text{swing}} = 30$. We initialize $\phi_1 = 0, \phi_4 = 0$ and $\phi_2 = 30, \phi_3 = 30$ so the swing phase for nondiagonal legs are offset by half a cycle. Similarly, we can realize a walking gait with $T = 120, T_{\text{swing}} = 30$, and initialize $\phi_1 = 0, \phi_2 = 90, \phi_3 = 60, \phi_4 = 30$.

Control policies can be learned with the centroidal models of Laikago and A1 in 1–2 hours for trotting (up to $0.5\,\text{m/s}$) and walking (up to $0.2\,\text{m/s}$). These policies are then deployed directly on their respective full-order robot models. We compare the behaviors of the simplified model and full model in Fig. 4. While the exact trajectories differ, they indicate similar general behavior.

As an alternative to the learned policy, we implement a baseline policy consisting of a PD controller used in [3, 4, 5] for the robot; specifically, the desired acceleration a_d is computed as

$$\begin{bmatrix} \ddot{p}_d \\ \dot{\omega}_d \end{bmatrix} = k_p \begin{bmatrix} p_d - p \\ \Theta_d - \Theta \end{bmatrix} + k_d \begin{bmatrix} \dot{p}_d - \dot{p} \\ \omega_d - \omega \end{bmatrix}, \tag{5}$$

with $p = [0, 0, p_{z,d}], \dot{p} = [\dot{p}_{x,d}, 0, 0],\ \Theta_d = [0, 0, 0], \omega_d = [0, 0, 0]$ being the desired behavior while k_p and k_d are diagonal matrices defining the PD gain. Note that the proportional gains for the x and y positions are both zero since these can be difficult to estimate on the physical robot.

Both the learned policy and the PD-based baseline policy can produce the trotting gaits and walking gaits on flat ground. In Fig. 5, we plot $Mf - a$, the error term for the QP problem (Equation 3), for both the learned policy and the PD-based policy with a trotting gait. Due to the underactuation of the system, the error is unavoidable, but the learned policy incurs much smaller error compared to a PD-based policy. We will observe later in this section that the PD-based policy also fails to solve more challenging tasks such as stepping stone and balance beam traversal, as well as two-legged balancing.

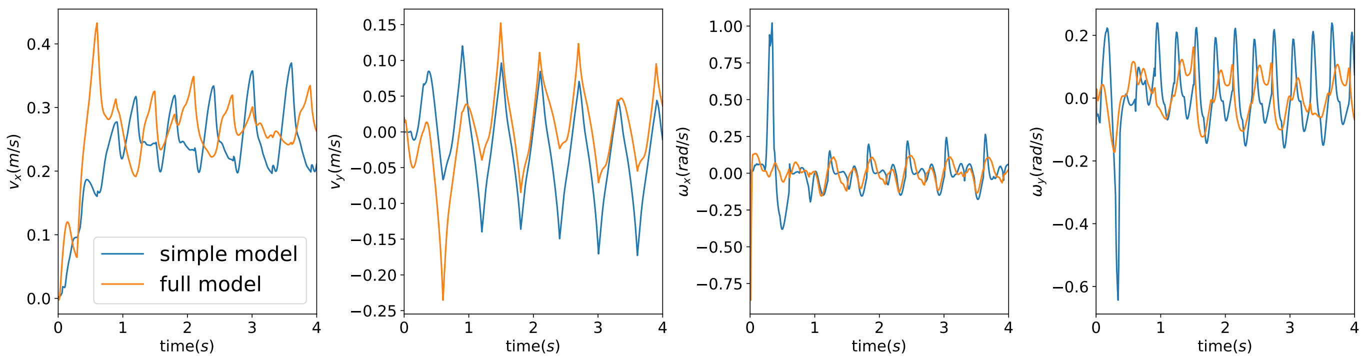


Fig. 4: Comparison of full-order model and simplified model of Laikago with a trotting policy. We plot the linear velocity on the x-y plane as well as the roll and pitch velocity. The general behaviors of the centroidal model and full model are close.

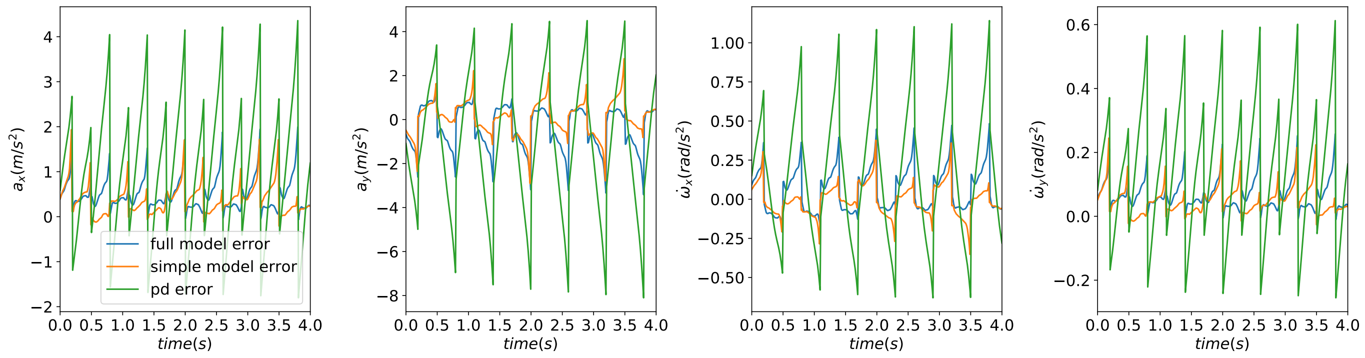


Fig. 5: Comparison of difference between the desired body acceleration and the actual body acceleration. Due to the underactuation of the systems, the non-zero difference is unavoidable. We plot the error of the learned policy for the simplified model and the full-order model, and plot the error of a PD policy on the full-order model. The learned policy produces acceleration commands with a much smaller discrepancy compared to a naive PD-based policy.

Since we are learning with a centroidal model, the policy is agnostic to the morphology of the robot. We test the learned policy on a version of Laikago where the hind-leg knee configuration is inverted. The learned policy works directly for the inverted-configuration robot. We also train a trotting policy for the default Laikago, utilizing the full physics simulation and directly generate commands at the joint control level, similar to [6, 7]. This end-to-end trained policy works well with the default configuration of Laikago, as expected, but fails to generalize to the inverted configuration due to the learned control policy being very specific to the morphology it was trained on.

B. Locomotion across Stepping Stones

To showcase that the method is compatible with tasks that require perception of the terrain, we train policies to drive the centroidal model across stepping stones. This terrain restricts the set of valid stepping locations, so the foot placement strategy needs to avoid infeasible locations on the ground. This problem has been considered for quadrupedal locomotion in recent work using model-based control approaches for trotting [41] and static walking [42], and a simpler version of the problem has been considered using a learning-based approach [11]. We demonstrate the generality of our approach by solving the stepping stone problem with both Laikago and A1, with either a trotting gait or a walking gait, including with turns.

Fig. 6 illustrates the stepping stone problem. As before, we use the Raibert-style heuristic to plan foot placement. However, if the planned foot placement is infeasible, the closest feasible position is chosen instead, as modeled by a $1\,\mathrm{m} \times 1\,\mathrm{m}$ grid of samples centered on the robot. We train a policy to traverse random stepping stone terrains with feasible footholds being placed between 10 cm to 20 cm apart for Laikago and 5 cm to 15 cm apart for A1. The training takes 4 hours, and the resulting policy can be realized on the simulated full model of Laikago and A1.

We compare the learned policy with the baseline PD-based control policy. The PD-based control policy can be tuned to navigate across the stepping stone terrain with $T = 40, T_{\text{swing}} = 20$ with a trotting gait on Laikago, which is unrealistically fast for the hardware. We failed to find a baseline PD-control policy that achieves a more realistic stepping frequency with $T = 60, T_{\text{swing}} = 30$. We believe the failure is caused by aggressive tracking of the command, whereas the learned policy prioritizes safety over tracking error.

The policy also generalizes to stepping stone patterns not seen during training. We command the robot to walk across the field of stepping stones while turning, while the trained

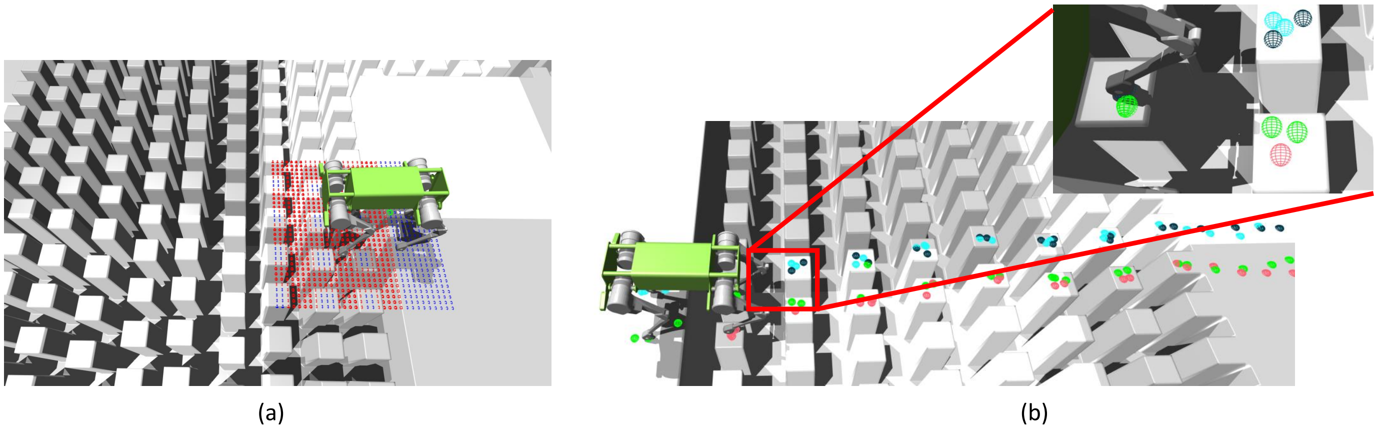


Fig. 6: Illustration of the stepping stone problem. (a) The system gets a local map of the terrain around the robot, illustrated by the grid of red and blue points. Blue points indicate regions suitable for foot placement while the red points indicate otherwise. The edges of the stepping stones are also deemed unsuitable to encourage safety. The foot placement strategy chooses the feasible point that is closest to the default foot placement, as given by the Raibert heuristic. (b) Footstep pattern of Laikago trotting across a stepping stone terrain. Different colors indicate stepping locations of different feet. (Please see video for further details)

policy has only seen straight-line forward motion over stepping stones. The policy successfully generalizes to this scenario.

C. Balance Beam

While a well-tuned baseline PD controller can solve some of the aforementioned tasks, it fails for tasks such as that shown in Fig. 3(c), due to the system underactuation and the greedy nature of a PD controller. We choose to solve the balance beam tasks to showcase the ability of our system to make control decisions that optimize performance over a time horizon. A similar capability was also demonstrated in [43] using model-based method.

In the balance beam scenario, the robot must walk on a narrow path with limited foot placement choice in the lateral direction. We train a balance beam policy for the centroidal model of the A1. During training, we restrict the foot placement in world coordinates with $|p_{i,y}| \leq 0.05\,\mathrm{m}$ while the nominal leg distance between left and right foot is 0.2 m. In addition, the y-coordinate of the rigid body p_y is also provided to the policy to avoid drift. We train a trotting policy with desired velocity $\dot{p}_{x,d} = 0.1$ m/s. The learned policy successfully transfers to the full simulated model of the A1 trotting across a balance beam.

D. Two-Legged Balancing

Similar to the balance beam task, two-legged balancing is also not achievable with a PD controller because the system has little control authority in the lateral direction. A naive PD controller quickly results in a fall to one side. This is also more challenging than the balancing beam task since the robot cannot regain balance by taking steps. In the two-legged balancing scenario, we set $\phi_1, \phi_4 = 31$ and $\phi_2, \phi_3 = 0$ for all time such that the left front leg and the right rear leg are set to be the stance legs. We also include p_x, p_y as input to the policy. With the full-order model, the right front leg and the left hind leg are held in place in midair using position control, and the stance legs are controlled in the same way as before.

It is worth noting that the performance for the balancing task is more sensitive to the body inertia as compared to other tasks. Since the inertia we use for the simplified model is an approximation of the full-order model parameters, a learned policy trained with the default inertia parameter fails on the full order model. We employ dynamics randomization, a technique frequently relied upon in RL training of quadrupedal locomotion [8, 9, 20, 7] and randomize the inertia parameters by 50 to 150 percent of the default values. The learned policy trained with the randomized centroidal model of the A1 is able to achieve two-legged balancing on the full A1 model.

E. Sim-to-Real Tests

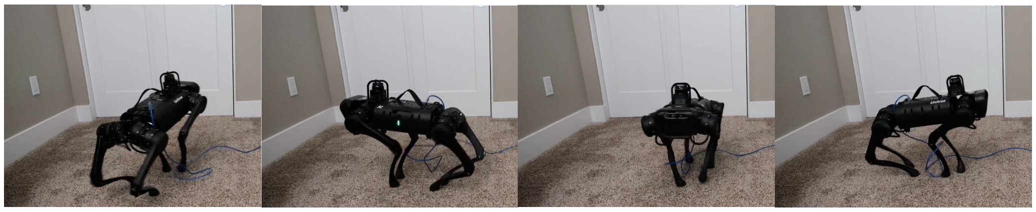
Finally, we transfer some of the policies to the physical A1 robot.[2] The control policy does not rely on a full-body physics simulator and thus avoids the most common types of overfitting problems. Sim-to-real transfer is as straightforward as the reduced-order to full model transfer. This further demonstrates the generality and robustness of our approach. Some snapshots of the sim-to-real tests can be seen in Fig. 7.

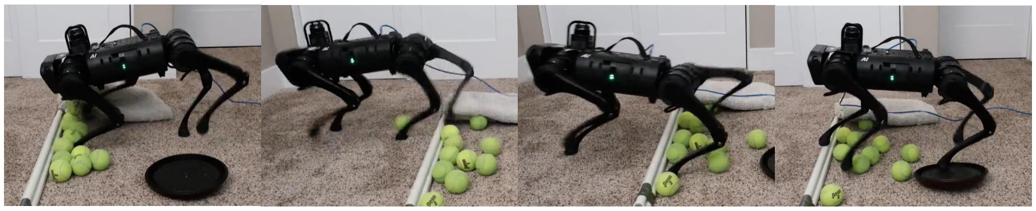
To generate gaits of different styles, we train a trotting policy on flat terrain with the default foot position $r_{\text{ref},x} = \pm 0.1\,\mathrm{m}, r_{\text{ref},y} = \pm 0.05\,\mathrm{m}$, twice as close to the center of mass as the default. We further set $T = 40$ and $T_{\text{swing}} = 20$ so the policy is trotting at a higher frequency. This policy can transfer to the full-order model in simulation and directly transfer to the physical A1. Fig. 7(a) shows some snapshots of this policy turning in place.

The learned policies are also robust to perturbation. In Fig. 7(b) we show some snapshots of a physical robot handling obstacle-strewn terrain using a policy trained only on flat terrain, with normal foot spacing and stepping frequency.

While we train policies to traverse stepping stones, we are not yet able to realize this on hardware due to our current lack of a working perception system. Instead, we design a

[2]Due to COVID-19 restrictions, we were unable to access the experimental space needed to perform experiments using the larger Laikago robot. We thus focus the sim-to-real tests on the more compact A1 robot.


(a)


(b)

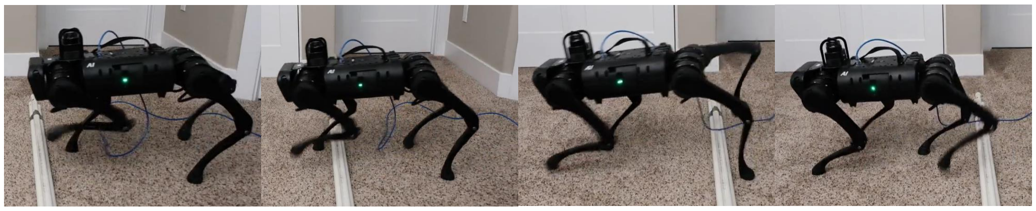

(c)

Fig. 7: Snapshots of sim-to-real tests for GLiDE. (a) A flat terrain policy commanded to spin around on the physical robot. (b) A flat terrain policy is tested for robustness to unstructured terrain with unmodelled obstancles of various types. (c) A stepping stone policy is commanded to step across a virtual gap made of a horizontal bar in physical environment. Please see video for further details.

proxy test where a virtual gap 0.2 m wide is placed in front of the robot (Fig. 7(c)). The foot placement algorithm is then modified to generate footholds that avoid this virtual gap. The stepping stone policy is able to generate commands that enable the stance legs to stabilize the robot while the swing legs take larger than usual steps.

V. Discussion and Future Work

We present a framework that combines reinforcement learning with a centroidal model to learn locomotion policies for quadrupedal locomotion. We validate our approach by efficiently learning locomotion policies for different quadrupedal robots and solving challenging tasks such as locomotion across stepping stones, two-legged balancing, and locomotion across a balance beam. We also demonstrate successful sim-to-real transfer. We believe this opens the door for efficient learning of skills for legged robots.

There remain constraints that are not explicitly captured by the centroidal model, including joint limits and torque limits. These could be taken into account implicitly, by using the full model to advance the state of the centroidal model, with the caveat that the learned control policy then becomes specific to the given full model. Obtaining maximal performance may also require more explicitly taking these limits into account, as can in principal be achieved using full-model RL.

While we obtain efficient training when compared to methods relying on full-model physics simulation on a CPU, we currently still lag in terms of wall-clock time compared to reinforcement learning work that fully utilizes GPU simulation, e.g., [24]. This is partly due to our naive implementation of the QP solver, which takes more than 80 percent of the simulation time. We expect that performance could be improved by one to two orders of magnitude with an optimized GPU-accelerated batched QP solver.

In the recent model-based control literature, highly dynamic motions such as flipping and bounding can be achieved with a centroidal model. For our current locomotion tasks, we focus on the common trotting and walking gaits. We believe that it should be feasible to learn policies for more general gait patterns and more dynamic motions.

Currently we rely on a simple heuristic for foot placement. This limits our ability to deal with more challenging terrain, e.g., more challenging variations of the stepping stone task. Incorporating systems [36, 44] that directly learn foot placement will be crucial for further improving the capability and robustness of our system.

Our system can in principle also handle terrains with height variations, e.g., for climbing stairs. We are in the process of extending our framework for learning to traverse more general terrains. We are also in the process of incorporating a vision system onto the physical A1 robot to enable sim-to-real transfer for tasks such as stepping stone traversal.

Currently we train different policies for a centroidal model with different parameters. It will be interesting to train a unified policy that is conditioned on model parameters and gait patterns, similar to what is done in [45, 46]. Such a policy should then generalize to all possible variations of quadrupeds by performing online adaption, similar to [6, 46].

References

[1] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the mit cheetah 3 through convex model-predictive control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–9.

[2] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8484–8490.

[3] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[4] 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," *arXiv preprint arXiv:2009.10019*, 2020.

[5] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepftinger, and R. Y. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *IEEE International Conference on Robotics and Automation (ICRA), 2013: 6-10 May 2013, Karlsruhe, Germany*. IEEE, 2013, pp. 3287–3292.

[6] 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.

[7] Z. Xie, X. Da, M. van de Panne, B. Babich, and A. Garg, "Dynamics randomization revisited: A case study for quadrupedal locomotion," *arXiv preprint arXiv:2011.02404*, 2020.

[8] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[9] 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.

[10] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, 2020. [Online]. Available: https://robotics.sciencemag.org/content/5/47/eabc5986

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

[12] Y. Gong and J. Grizzle, "Angular momentum about the contact point for control of bipedal locomotion: Validation in a lip-based controller," *arXiv preprint arXiv:2008.10763*, 2020.

[13] K. Green, Y. Godse, J. Dao, R. L. Hatton, A. Fern, and J. Hurst, "Learning spring mass locomotion: Guiding policies with a reduced-order model," *arXiv preprint arXiv:2010.11234*, 2020.

[14] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1991, pp. 1405–1406.

[15] 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.

[16] X. Xiong and A. Ames, "3d underactuated bipedal walking via h-lip based gait synthesis and stepping stabilization," *arXiv preprint arXiv:2101.09588*, 2021.

[17] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1517–1524, 2019.

[18] G. Bledt and S. Kim, "Extracting legged locomotion heuristics with regularized predictive control," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 406–412.

[19] K. Paigwar, L. Krishna, S. Tirumala, N. Khetan, A. Sagi, A. Joglekar, S. Bhatnagar, A. Ghosal, B. Amrutur, and S. Kolathaya, "Robust quadrupedal locomotion on sloped terrains: A linear policy approach," *arXiv preprint arXiv:2010.16342*, 2020.

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

[21] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, 2020.

[22] H. Bharadhwaj, A. Kumar, N. Rhinehart, S. Levine, F. Shkurti, and A. Garg, "Conservative safety critics for exploration," in *International Conference on Learning Representations (ICLR)*, apr 2021.

[23] E. Heiden, D. Millard, E. Coumans, Y. Sheng, and G. S. Sukhatme, "Neuralsim: Augmenting differentiable simulators with neural networks," *arXiv preprint arXiv:2011.04217*, 2020.

[24] NVIDIA, *Isaac Gym - Preview Release*, 2020. [Online]. Available: https://developer.nvidia.com/isaac-gym

[25] D. Jain, A. Iscen, and K. Caluwaerts, "From pixels to legs: Hierarchical learning of quadruped locomotion," *arXiv preprint arXiv:2011.11722*, 2020.

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

[27] T. Li, R. Calandra, D. Pathak, Y. Tian, F. Meier, and A. Rai, "Planning in learned latent action spaces for generalizable legged locomotion," *arXiv preprint arXiv:2008.11867*, 2020.

[28] 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 *IEEE International Conference on Intelligent Robots and Systems (IROS)*, nov 2019. [Online]. Available: https://stanfordvl.github.io/vices/

[29] R. Martin-Martin, A. Allshire, C. Lin, S. Mendes, S. Savarese, and A. Garg, "LASER: Learning a Latent Action Space for Efficient Reinforcement Learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2021. [Online]. Available: https://www.pair.toronto.edu/laser/

[30] J. Carius, F. Farshidian, and M. Hutter, "Mpc-net: A first principles guided policy search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, 2020.

[31] X. Da and J. Grizzle, "Combining trajectory optimization, supervised machine learning, and model structure for mitigating the curse of dimensionality in the control of bipedal robots," *The International Journal of Robotics Research*, vol. 38, no. 9, pp. 1063–1097, 2019.

[32] J. Viereck and L. Righetti, "Learning a centroidal motion planner for legged locomotion," *arXiv preprint arXiv:2011.02818*, 2020.

[33] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5280–5286.

[34] T. Li, H. Geyer, C. G. Atkeson, and A. Rai, "Using deep reinforcement learning to learn high-level policies on the atrias biped," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 263–269.

[35] T. Li, K. Srinivasan, M. Q.-H. Meng, W. Yuan, and J. Bohg,

"Learning hierarchical control for robust in-hand manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8855–8862.

[36] A. Meduri, M. Khadiv, and L. Righetti, "Deepq stepper: A framework for reactive dynamic walking on uneven terrain," *arXiv preprint arXiv:2010.14834*, 2020.

[37] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.

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

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[40] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[41] R. Grandia, A. J. Taylor, A. D. Ames, and M. Hutter, "Multi-layered safety for legged robots via control barrier functions and model predictive control," *arXiv preprint arXiv:2011.00032*, 2020.

[42] C. Mastalli, I. Havoutis, M. Focchi, D. G. Caldwell, and C. Semini, "Motion Planning for Quadrupedal Locomotion: Coupled Planning, Terrain Mapping and Whole-Body Control," *IEEE Transactions on Robotics (T-RO)*, 2020.

[43] C. Gonzalez, V. Barasuol, M. Frigerio, R. Featherstone, D. G. Caldwell, and C. Semini, "Line walking and balancing for legged robots with point feet," *arXiv preprint arXiv:2007.01087*, 2020.

[44] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *ArXiv*, vol. abs/2012.03094, 2020.

[45] J. Won and J. Lee, "Learning body shape variation in physics-based characters," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 6, pp. 1–12, 2019.

[46] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.