Learning Agile Locomotion and Manipulation with Legged Robots

Xuxin Cheng

CMU-RI-TR-23-60

August, 2023

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

To my family.

Abstract

Legged robotics has seen significant advancements in locomotion. However, there remain significant gaps compared to their biological counterparts, particularly in energy efficiency, natural motion, and the capacity for agile skills. This thesis primarily focuses on two aspects: the unified control of legged manipulators and the development of novel control algorithms for multi-skill quadrupeds.

The first study presents a strong counter to the standard hierarchical control pipeline for legged manipulators, which is characterized by immense engineering to support coordination between the arm and legs, often resulting in non-smooth unnatural motions. In this work, we propose to learn a unified policy for whole-body control of a legged manipulator using reinforcement learning. We propose Regularized Online Adaptation to bridge the Sim2Real gap for high-DoF control, and Advantage Mixing exploiting the causal dependency in the action space to overcome local minima during training the whole-body system. We also present a simple design for a low-cost legged manipulator, and find that our unified policy can demonstrate dynamic and agile behaviors across several task setups.

The second study dives further into the field where robotic quadrupeds are still far behind their biological counterparts, such as dogs, which display a variety of agile skills and can use the legs beyond locomotion to perform several basic manipulation tasks like interacting with objects and climbing. We train quadruped robots not only to walk but also to use the front legs to climb walls, press buttons, and perform object interaction in the real world. To navigate this challenging optimization, we decouple the skill learning broadly into locomotion, involving movement whether via walking or climbing a wall, and manipulation, involving using one leg to interact while balancing on the other three legs. We also devise a behavior tree that encodes a high-level task hierarchy from one clean expert demonstration, thereby combining these skills into a robust long-term plan. Finally, we apply a sim2real variant that builds upon recent locomotion success to transfer these skills to the real world. Evaluations in both simulation and real-world settings exhibit successful executions of both short and long-range tasks, underscoring the robustness confronting external perturbations.

Altogether, these studies signify substantial progress towards more dexterous, efficient, and robust legged robots.

Acknowledgments

First, I would like to express my deepest appreciation to my advisor, Prof. Deepak Pathak. His unwavering support throughout my master's journey has been nothing short of remarkable. His insights during our one-on-one meetings and guidance on conducting top-notch research have been invaluable.

I am extremely grateful to my thesis committee members, Prof. Abhinav Gupta and Tianyi Zhang for their helpful feedback.

I wish to acknowledge my collaborators, Zipeng Fu and Ashish Kumar. My collaboration with Zipeng on the initial project was a crucial learning experience and laid a strong foundation for my subsequent projects. I also want to express my gratitude to Ashish for his guidance during the second project, and for representing our work at the conference when I couldn't attend in person.

It is also an honor to have fun with lovely and supportive freinds around CMU. I would like to extend my thanks to Heng Yu, Haolun Zhang, Jinqi Luo, Zixuan Huang, Tianyuan Zhang, Haoyu Xiong, Carl Qi, and Jianren Wang. In particular, I appreciate all the members of LEAP Lab for making my master's journey memorable.

Finally, I want to say a heartfelt thank you to my family. My parents, Junxia Cheng and Guoqiang Cheng, have given me love and support at every turn, and for that, I am forever grateful. To my girlfriend, Ashley Zhang, your unwavering love and support have been a source of strength throughout this journey.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

Locomotion has seen impressive performance in the last decade with results in challenging outdoor and indoor terrains, otherwise unreachable by their wheeled or tracked counterparts. However, there are strong limitations to what a legged-only robot can achieve since even the most basic everyday tasks, besides visual inspection, require some form of manipulation.

This has led to widespread interest and progress towards building legged manipulators, i.e., robots with both legs and arms, primarily achieved so far through physical modeling of dynamics. However, modeling a legged robot with an attached arm is a dynamic, high-DoF, and non-smooth control problem, requiring substantial domain expertise and engineering effort on the part of the designer. The control frameworks are often modular where kinematic constraints are dealt with separately for different control spaces, thus limited to operating in constrained settings with limited generalization. Learning-based methods, such as reinforcement learning (RL), could help lower the engineering burden while aiding generalization to diverse scenarios.

While the attached arm can greatly expand the agility of legged robots, their biological counterparts can already use their legs more generally to achieve some manipulation tasks, without the extra burden of the arm. Indeed, locomotion and manipulation can be seen as dual of each other and share evolutionary origins where frontal legs evolved to become arms in bipeds. Providing robotic quadrupeds a similar ability would not only push the *agility* of robotic locomotion but can also greatly expand the reach as well as functionality even if there is an arm already attached on top of the quadruped.

In this thesis, we focus on expanding the agility of quadruped robots with manipulation capabilities by both adding an attached arm and utilizing quadrupeds' own frontal limbs.

- In Chapter 2, we present both a hardware setup for customized low-cost fully untethered legged manipulators and a method for learning one unified policy to control and coordinate both legs and arm, which is compatible with diverse operating modes. We use our unified policy for whole-body control, i.e. to control the joints of the quadruped legs as well as the manipulator to simultaneously take the arm end-effector to desired poses and command the quadruped to move in desired velocities. The key insights of the method are that we can exploit the causal structure in action space with respect to manipulation and locomotion to stabilize and speed up learning, and adding regularization to domain adaptation bridges the gap between simulation with full states and real world with only partial observations.
- In Chapter 3, we focus on this joint problem of learning locomotion as well as basic manipulation skills by expanding the capability of quadruped robots to enable them to use their legs as manipulators. In particular, we focus on tasks like climbing a wall with front legs, jumping on a wall to reach a button, using a leg to push a button, etc; and then combining them to achieve long-range behaviors. More broadly, we follow the popular approach of learning environment latent conditioned policies in simulation using reinforcement learning (RL) and then transferring them to the real world via sim2real.
- In Appendix A, we provide additional details for Deep Whole-Body Control about Regularized Online Adaptation, Advantage Mixing, simulation and training, and real-world experiments.

Chapter 2

Deep Whole-Body Control: Learning a Unified Policy for Manipulation and Locomotion

2.1 Introduction

Locomotion has seen impressive performance in the last decade with results in challenging outdoor and indoor terrains, otherwise unreachable by their wheeled or tracked counterparts. However, there are strong limitations to what a legged-only robot can achieve since even the most basic everyday tasks, besides visual inspection, require some form of manipulation. This has led to widespread interest and progress towards building legged manipulators, i.e., robots with both legs and arms, primarily achieved so far through physical modeling of dynamics [24, 27, 59, 70, 79, 89]. However, modeling a legged robot with an attached arm is a dynamic, high-DoF, and non-smooth control problem, requiring substantial domain expertise and engineering effort on the part of the designer. The control frameworks are often hierarchical where kinematic constraints are dealt with separately for different control spaces [74], thus limited to operating in constrained settings with limited generalization. Learning-based methods, such as reinforcement learning (RL), could help lower the engineering burden while aiding generalization to diverse scenarios.

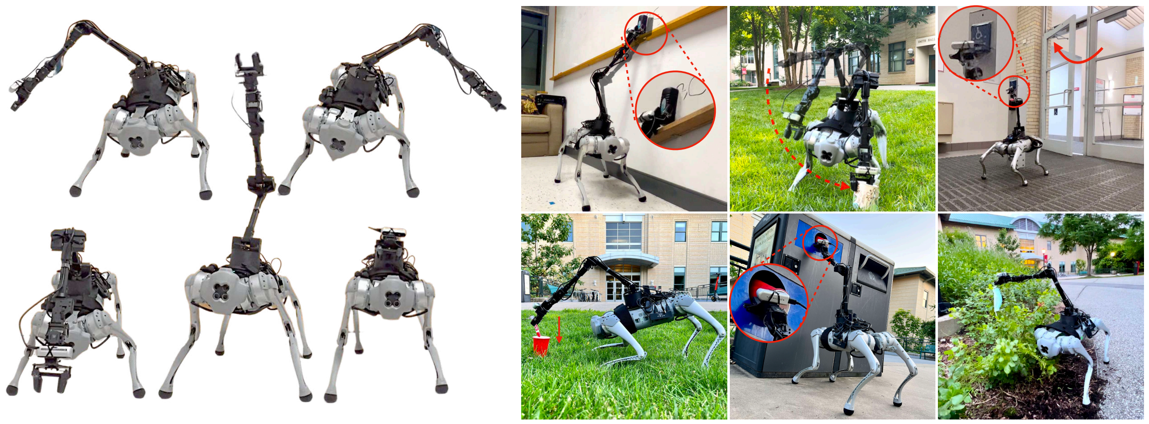


Figure 2.1: We present a framework for whole-body control of a legged robot with a robot arm attached. Left half shows how whole-body control achieves larger workspace by leg bending and stretching. Right half shows different real-world tasks, including wiping whiteboard, picking up a cup, pressing door-open buttons, placing, throwing a cup into a garbage bin and picking in clustered environments.

However, recent learning-based approaches for legged mobile manipulators [51] have also followed their model-based counterparts [8, 95] by using hierarchical models in a semi-coupled fashion to control the legs and arm. This is ineffective due to several practical reasons including lack of coordination between the arm and legs, error propagation across modules, and slow, non-smooth and unnatural motions. Furthermore, it is far from the whole-body motor control in humans where studies suggest strong coordination among limbs. In fact, the control of hands and legs is so tied together that they form low-dimension synergies, as outlined over 70 years ago in a seminal series of writings by Russian physiologist Nikolai Bernstein [9, 12, 41]. Perhaps the simplest example is how it is hard for humans to move one arm and the corresponding leg in different motions while standing. The whole-body control should not only allow coordination but also extend the capabilities of the individual parts. For instance, our robot bends or stretches its legs with the movement of the arm to extend the reach of the end-effector as shown in Figure 2.1.

Unlike legged locomotion, it is not straightforward to scale the standard sim2real RL to whole-body control due to several challenges: (a) *High-DoF control*: Our robot shown in Figure 2.3 has total 19 degrees of freedom. This problem is exacerbated in legged manipulators because the control is dynamic, continuous and high-frequency,

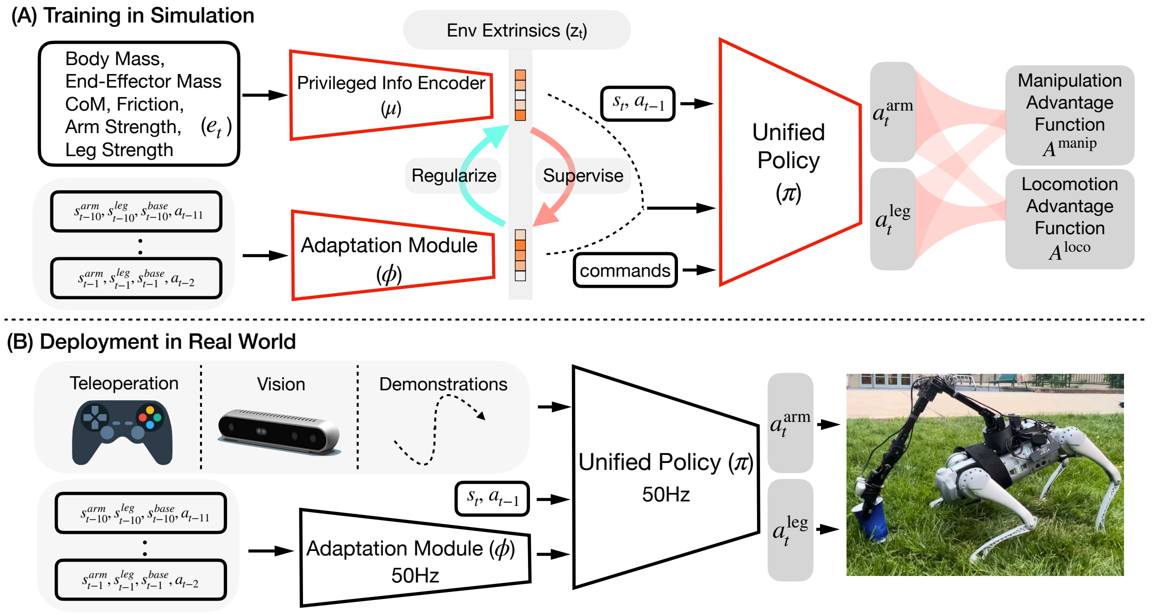

(A) Training in Simulation
Body Mass, End-Effector Mass CoM, Friction, Arm Strength, Leg Strength (e_t)
Privileged Info Encoder (μ)
Env Extrinsics (z_t)
Regularize
Supervise
Adaptation Module (ϕ)
$s^{arm}_{t-10}, s^{leg}_{t-10}, s^{base}_{t-10}, a_{t-11}$
$s^{arm}_{t-1}, s^{leg}_{t-1}, s^{base}_{t-1}, a_{t-2}$
s_t, a_{t-1}
commands
Unified Policy (π)
a^{arm}_t
a^{leg}_t
Manipulation Advantage Function A^{manip}
Locomotion Advantage Function A^{loco}
(B) Deployment in Real World
Teleoperation
Vision
Demonstrations
Unified Policy (π) 50Hz
Adaptation Module (ϕ) 50Hz

Figure 2.2: Whole-body control framework. During training, a unified policy is learned by conditioned on environment extrinsics. During deployment, the adaptation module is reused without any real-world fine-tuning. The robot can be commanded in various modes including teleoperation, vision and demonstration replay.

which leads to an exponentially large search space even in few seconds of trajectory. (b) *Conflicting objectives and local minima*: Consider when the arm tilts to the right, the robot needs to change the walking gait to account for the weight balance. This curbs the locomotion abilities and makes training prone to learn only one mode (manipulation or locomotion) well. (c) *Dependency*: Consider picking an object on the ground, the end-effector of the arm needs support from the torso by bending legs. This means the absolute performance of manipulation is bounded until legs can adapt.

In this work, we present both a hardware setup for customized low-cost fully untethered legged manipulators and a method for learning one unified policy to control and coordinate both legs and arm, which is compatible with diverse operating modes as shown in Figure 2.1. We use our unified policy for whole-body control, i.e. to control the joints of the quadruped legs as well as the manipulator to simultaneously take the arm end-effector to desired poses and command the quadruped to move in desired velocities. The key insights of the method are that we can exploit the causal structure in action space with respect to manipulation and locomotion to stabilize and speed up learning, and adding regularization to domain adaptation bridges the gap between simulation with full states and real world with only partial observations.

We perform evaluation on our proposed legged manipulator. Despite immense progress, there exists no easy-to-use legged manipulator for academic labs. Most publicized robot is Spot Arm from Boston Dynamics [18], but the robot comes with pre-designed controllers that cannot be changed. Another example is the ANYmal robot with a custom arm [51] from ANYBotics. Notably, both these hardware setups are expensive (more than 100K USD). We implement a simple design of low-cost legged Go1 robot [82] with low-cost arm on top (hardware costs 6K USD). Our legged manipulator can run fully untethered with modest on-board compute. We show the effectiveness of our learned whole-body controller for teleoperation, vision-guided control as well as open-loop control setup across tasks such as picking objects, throwing garbage, pressing buttons on walls etc. Our robot exhibits **dynamic** and **agile** leg-arm coordinated motions as shown in videos at https://maniploco.github.io.

2.2 Method: A Unified Policy for Coordinated Manipulation and Locomotion

We formulate the unified policy π as one neural network where the inputs are current base state $s_t^{\text{base}} \in \mathbb{R}^5$ (row, pitch, and base angular velocities), arm state $s_t^{\text{arm}} \in \mathbb{R}^{12}$ (joint position and velocity of each arm joint), leg state $s_t^{\text{leg}} \in \mathbb{R}^{28}$ (joint position and velocity of each leg joint, and foot contact indicators), last action $a_{t-1} \in \mathbb{R}^{18}$, end-effector position and orientation command $[p^{\text{cmd}}, o^{\text{cmd}}] \in \mathbb{SE}(3)$, base velocity command $[v_x^{\text{cmd}}, \omega_{\text{yaw}}^{\text{cmd}}]$, and environment extrinsics $z_t \in \mathbb{R}^{20}$ (details in Section 2.2.2). The policy outputs target arm joint position $a_t^{\text{arm}} \in \mathbb{R}^6$ and target leg joint position $a_t^{\text{leg}} \in \mathbb{R}^{12}$, which are subsequently converted to torques using PD controllers. We use joint-space position control for both legs and the arm. As opposed to operational space control of the arm, joint-space control enables learning to avoid self-collision and smaller Sim-to-Real gap, which is also found to be useful in other setups involving multiple robot parts, like bimanual manipulation [35].

We use RL to train our policy π by maximizing the discounted expected return

	Command Following ($r_{\text{following}}$)	Energy (r_{energy})	Alive (r_{alive})
r^{manip}	$0.5 \cdot e^{-\lVert [p,o]-[p^{\text{cmd}},o^{\text{cmd}}] \rVert_1}$	$-0.004 \cdot \sum_{j \in \text{arm joints}} \lvert \tau_j \dot{q}_j \rvert$	0
r^{loco}	$-0.5 \cdot \left\lvert v_x - v_x^{\text{cmd}} \right\rvert + 0.15 \cdot e^{-\left\lvert \omega_{\text{yaw}} - \omega_{\text{yaw}}^{\text{cmd}} \right\rvert}$	$-0.00005 \cdot \sum_{i \in \text{leg joints}} \lvert \tau_i \dot{q}_i \rvert^2$	$0.2 + 0.5 \cdot v_x^{\text{cmd}}$

Table 2.1: Both manipulation and locomotion rewards follow: $r_{\text{following}} + r_{\text{energy}} + r_{\text{alive}}$, which encourages command following while penalizes positive mechanical energy consumption to enable smooth motion [21]. Denote forward base linear velocity v_x, yaw angular base velocity ω_{yaw}, torque τ, joint angle velocity $\dot{q}$.

Command Vars	Training Ranges	Test Ranges
v_x^{cmd}	[0, 0.9]	[0.8, 1.0]
$\omega_{yaw}^{\text{cmd}}$	[-1,0, 1.0]	[-1, -.7] & [.7, 1]
l	[0.2, 0.7]	[0.6, 0.8]
p	$[-2\pi/5, 2\pi/5]$	$[-2\pi/5, 2\pi/5]$
y	$[-3\pi/5, 3\pi/5]$	$[-3\pi/5, 3\pi/5]$
T_{traj}	[1, 3]	[0.5, 1]

Table 2.2: Ranges for uniform sampling of command variables

$\mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \right]$, where r_t is the reward at time step t, γ is the discount factor, and T is the maximum episode length. The reward r is the sum of manipulation reward r^{manip} and locomotion reward r^{loco} as shown in Table 2.1. Notice that we use the second power of energy consumption at each leg joint to encourage both lower average and lower variance across all leg joints. We follow the simple reward design that encourages minimizing energy consumption from [21].

Env Params	Training Ranges	Test Ranges
Base Extra Payload	[-0.5, 3.0]	[5.0, 6.0]
End-Effector Payload	[0, 0.1]	[0.2, 0.3]
Center of Base Mass	[-0.15, 0.15]	[0.20, 0.20]
Arm Motor Strength	[0.7, 1.3]	[0.6, 1.4]
Leg Motor Strength	[0.9, 1.1]	[0.7, 1.3]
Friction	[0.25, 1.75]	[0.05, 2.5]

Table 2.3: Ranges for uniform sampling of environment parameters

We parameterize the end-effector position command p^{cmd} in spherical coordinate (l, p, y), where l is the radius of the sphere and p and y are the pitch and yaw angle. The origin of the spherical coordinate system is set at the base of the arm, but independent of torso's height, row and pitch (details in Supplementary). We set the end-effector pose command p^{cmd} by interpolating between the current end-effector position p and a randomly sampled end-effector position p^{end} every T_{traj} seconds:

$$p_t^{\text{cmd}} = \frac{t}{T_{\text{traj}}} p + \left(1 - \frac{t}{T_{\text{traj}}}\right) p^{\text{end}},\ t \in [0, T_{\text{traj}}].$$

p^{end} is resampled if any p_t^{cmd} leads to self-collision or collision with the ground. o^{cmd} is uniformly sampled from $\mathbb{SO}(3)$ space. Table 3.1 lists the ranges for sampling of all command variables.

2.2.1 Advantage Mixing for Policy Learning

Training a robust policy for a high-DoF robot is hard. In both manipulation and locomotion learning literature, researchers have used curriculum learning to ease the learning process by gradually increasing the difficulty of tasks so that the policy can learn to solve simple tasks first and then tackle difficult tasks [4, 56, 71]. However, most of these works require many manual tunings of a diverse set of the curriculum parameters and careful design of the mechanism for automatic curriculum.

Instead of introducing a large number of curricula on the learning and environment setups, we rely on only one curriculum with only one parameter to expedite the policy learning. Since we know that manipulation tasks are mostly related to the arm actions and locomotion tasks largely depends on leg actions, we can formulate this inductive bias in policy optimization by mixing advantage functions for manipulation and locomotion to speed up policy learning. Formally, for a policy with diagonal Gaussian noise and a sampled transition batch D, the training objective with respect to policy's parameters θ_π is

$$J(\theta_\pi) = \frac{1}{|\mathcal{D}|} \sum_{(s_t, a_t) \in \mathcal{D}} \log \pi(a_t^{\text{arm}} \mid s_t) \left(A^{\text{manip}} + \beta A^{\text{loco}}\right) + \log \pi(a_t^{\text{leg}} \mid s_t) \left(\beta A^{\text{manip}} + A^{\text{loco}}\right)$$

β is the curriculum parameter that linearly increases from 0 to 1 over timesteps

T_{mix}: $\beta = \min(t/T_{\text{mix}}, 1)$. A^{manip} and A^{loco} are advantage functions based on r^{manip} and r^{loco} respectively. Intuitively, the Advantage Mixing reduces the credit assignment complexity by first attributing difference in manipulation returns to arm actions and difference in locomotion returns to leg actions, and then gradually anneal the weighted advantage sum to encourage learning arm and leg actions that help locomotion and manipulation respectively. We optimize this RL objective by PPO [73].

2.2.2 Regularized Online Adaptation for Sim-to-Real Transfer

Much prior work on Sim-to-Real transfer utilize the two-phase teacher-student scheme to first train a teacher network by RL using privileged information that is only available in simulation, and then the student network using onboard observation history imitates the teacher policy either in explicit action space or latent space [39, 43, 55, 58]. Due to the information gap between the full state available to the teacher network and partial observability of onboard sensories, the teacher network may provide supervision that is impossible for the student network to predict, resulting in a *realizability gap*. This problem is also noted in Embodied Agent community [83]. In addition, the second phase can only start after the convergence of the first phase, yielding extra burdens for both training and deployment.

To tackle the realizability gap and to remove the two-phase pipeline, we propose Regularized Online Adaptation (shown in Figure 2.2). Concretely, the encoder μ takes the privileged information e as input and predict an enviornment extrinsics latent z^{μ} for the unified policy to adapt its behavior in different environments. The adaptation module ϕ estimates the environment extrinsics latent z^{ϕ} by only condition on recent observation history from robot's onboard sensories. We jointly train μ with the unified policy π end-to-end by RL and regularize z^{μ} to avoid large deviation from z^{ϕ} estimated by the adaptation module. The adaption module ϕ is trained by imitating z^{μ} online. We formulate the loss function of the whole learning pipeline with respect to policy's parameters θ_{π}, privileged information encoder's parameters θ_{μ}, and adaptation module's parameters θ_{ϕ} as

$$L(\theta_{\pi}, \theta_{\mu}, \theta_{\phi}) = -J(\theta_{\pi}, \theta_{\mu}) + \lambda ||z^{\mu} - \text{sg}[z^{\phi}]||_2 + ||\text{sg}[z^{\mu}] - z^{\phi}||_2 \ ,$$

where $J(\theta_\pi, \theta_\mu)$ is the RL objective discussed in Section 2.2.1, $\text{sg}[\cdot]$ is the stop gradient operator, and λ is the Laguagrian multiplier acting as regularization strength. The loss function can be minimized by using dual gradient descent: $\theta_\pi, \theta_\mu \leftarrow \arg\min_{\theta_\pi, \theta_\mu} \mathbb{E}_{(s,a)\sim\pi(\dots,z^\mu)}[L]$, $\theta_\phi \leftarrow \arg\min_{\theta_\phi} \mathbb{E}_{(s,a)\sim\pi(\dots,z^\phi)}[L]$, and $\lambda \leftarrow \lambda + \alpha\frac{\partial L}{\partial \lambda}$ with step size α. This optimization process is known to converge under mild conditions [46, 81]. In practice, we alternate the optimization process of the unified policy π and encoder μ and the one of adaptation module ϕ by a fixed number of gradient steps. λ increases from 0 to 1 by a fixed linear scheme. Notice that RMA [39] is a special case of Regularized Online Adaptation, in which the Laguagrian multiplier λ is set to be constant zero and the adaptation module ϕ starts training only after convergence of the policy π and the encoder μ.

Deployment During deployment, the unified policy and adaptation module executes jointly onboard. To specify commands, we develop three interfaces: teleopertion by joysticks, closed-loop control by using RGB tracking, and open-loop reply of human demonstrations. Details are in Section 2.3.3.

2.3 Experimental Results

2.3.1 Robot System Setup

The robot platform is comprised of a Unitree Go1 quadraped [82] with 12 actuatable DoFs, and a robot arm which is the 6-DoF Interbotix WidowX 250s [1] with a parallel gripper. We mount the arm on top of the quadruped. The RealSense D435 provides RGB visual information and is mounted close to the gripper of WidowX. Both power of Go1 and WidowX are provided by Go1's onboard battery. Neural network inference is also done onboard of Go1. Our robot system uses only onboard computation and power so it is fully untethered.

2.3.2 Simulation Experiments

The purpose of our simulation experiments is to address the following questions:

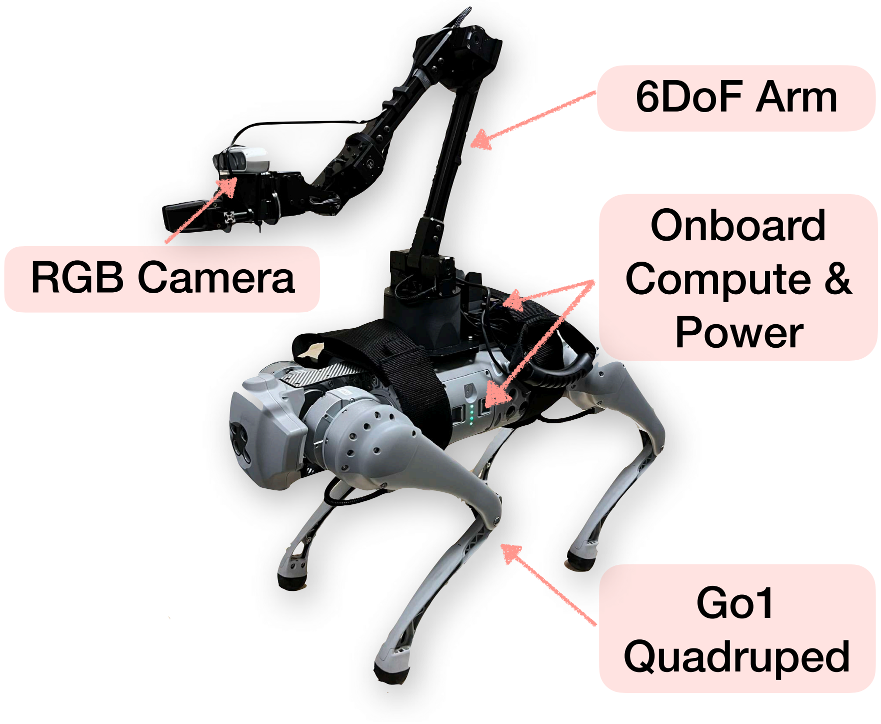


Figure 2.3: Robot system setup

- Does the unified policy improves over separate policies for the arm and legs? If so, how?
- How Advantage Mixing helps learning the unified policy?
- What's the performance of Regularized Online Adaptation compared with other Sim2Real methods?

	Survival ↑	Base Accel. ↓	Vel Error ↓	EE Error ↓	Tot. Energy ↓
Unified (Ours)	**97.1** $\pm$ 0.61	**1.00** $\pm$ 0.03	**0.31** $\pm$ 0.03	**0.63** $\pm$ 0.02	50 $\pm$ 0.90
Separate	92.0 $\pm$ 0.90	1.40 $\pm$ 0.04	0.43 $\pm$ 0.07	0.92 $\pm$ 0.10	51 $\pm$ 0.30
Uncoordinated	94.9 $\pm$ 0.61	1.03 $\pm$ 0.01	0.33 $\pm$ 0.01	0.73 $\pm$ 0.02	**50** $\pm$ 0.28

Table 2.4: Comparison of unified policy with separate policies for legs-arm, and one uncoordinated policy. The unified policy achieves the best performance given same energy consumption. The test ranges are in Table 3.1.

Baselines and Metrics: We compare our method with the following baselines:

	Arm workspace (m^3) ↑	Survival under perturb ↑
Unified (Ours)	**0.82** $\pm$ 0.02	**0.87** $\pm$ 0.04
Separate	0.58 $\pm$ 0.10	0.64 $\pm$ 0.06
Uncoordinated	0.65 $\pm$ 0.02	0.77 $\pm$ 0.06

Table 2.5: In unified policy, legs help increase the arm workspace and the arm helps the quadruped to stabilize.

1. Separate policies for legs and the arm: one policy controls legs based on the quadraped observation, and another policy controls the arm based on arm observation.
2. One uncoordinated policy: Same as unified policy which observes aggregate state of base, legs and the arm, but only r^{manip} is used to train arm actions, and only r^{loco} for leg actions.
3. Rapid Motor Adaptation (RMA) [39]: Two-phase teacher-student baseline.
4. Expert policy: the unified policy using the privileged information encoder z^{μ}.
5. Domain Randomization: the unified policy trained without environment extrinsics z.

We report following metrics: (1) survival percentage, (2) Base Accel: angular acceleration of base, (3) Vel Error: L1 error between base velocity commands and actual base velocity, (4) EE Error: L1 error between end-effector (EE) command and actual EE pose, (5) Tot. Energy: total energy consumed by legs and the arm. All metrics are normalized by episode length. All experiments are tested over 3 randomly initialized networks and 1000 episodes each. Details of simulation and training are in Supplementary.

Improvements of the Unified Policy over Baselines: In Table 2.4, our unified policy outperforms separate and uncoordinated policies because both the arm and leg actions are trained with the sum of reward for manipulation and locomotion are given with observations for the arm, legs and the quadraped base, while baselines struggle to maintain a small base acceleration, which results in larger error in command velocity following and inaccurate EE pose following.

Unified Policy Increases Whole-body Coordination: Table 2.5 shows that our unified policy promotes whole-body coordination where (1) leg actions will help the arm to achieve a larger workspace via bending for lower EE commands and standing up high for higher EE commands, and (2) arm will help the robot balance under larger perturbation (1.0 m/s initial velocity of base) resulting in higher survival rate of the unified policy. We estimate the the arm workspace via calculating the volume of the convex hull of 1000 sampled EE poses, subtracted by the volume of a cube that encloses the quadruped.

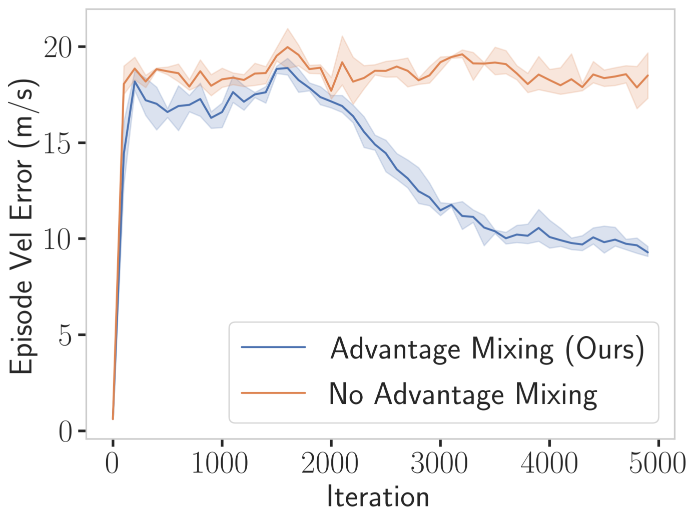


Figure 2.4: Advantage Mixing helps the unified policy to learn to follow the command velocity much faster (aggregated Vel Error over episodes decreases sharply) than without mixing.

Advantage Mixing Helps Learning the Unified Policy: Without Advantage Mixing, the unified policy has difficulty in credit assignment, resulting in the policy first learns EE command following but ignores the locomotion task. As shown in Figure 2.4, Advantage Mixing helps the policy to focus on each task first and then merge them together, which induces a curriculum-like mechanism to speed up training. Details in Supplementary.

Robust OOD Performance of Regularized Online Adaptation: We find that our Regularized Online Adaptation is more robust than RMA and Domain Random-

	Realizability Gap $\|z^\mu - z^\phi\|_2$ ↓	Survival ↑	Base Accel. ↓	Vel Error ↓	EE Error ↓	Tot. Energy ↓
Domain Randomization	-	95.8 ± 0.2	**0.44** ± 0.00	0.46 ± 0.00	0.40 ± 0.00	**21.9** ± 0.53
RMA [39]	0.31 ± 0.01	95.2 ± 0.2	0.54 ± 0.02	0.44 ± 0.00	0.26 ± 0.04	27.3 ± 0.95
Regularized Online Adapt (Ours)	**2e-4** ± 0.00	**97.4** ± 0.1	0.51 ± 0.02	**0.39** ± 0.01	**0.21** ± 0.00	25.9 ± 0.56
Expert w/ Reg.	-	97.8 ± 0.2	0.52 ± 0.02	0.40 ± 0.01	0.21 ± 0.00	25.8 ± 0.49
Expert w/o Reg.	-	98.3 ± 0.2	0.51 ± 0.02	0.39 ± 0.00	0.21 ± 0.00	25.6 ± 0.30

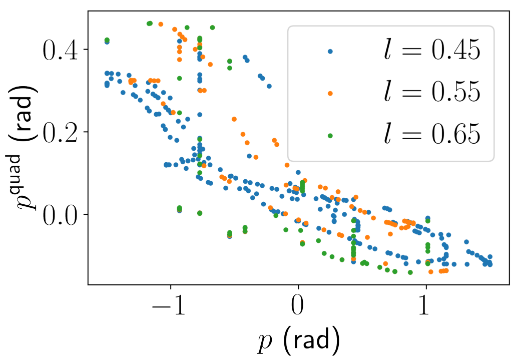
Table 2.6: Regularized Online Adaptation outperforms other baselines with the smallest imitation error which helps it to have the same performance as the expert policy which uses privileged information to predict environment extrinsics. Expert policy trained with regularization term $\|z^\mu - \text{sg}[z^\phi]\|_2$ has negligible performance degradation compared with the expert trained without regularization. Test ranges in Table 2.3. Domain Randomization learns to just stand in most cases, hence, trivially collapsing to low Tot. Energy and Base Accel.

ization (DR), tested in environments with out-of-distribution (OOD) environment parameters in Table 2.3. In RMA, it is not guaranteed the estimated environment extrinsics by the adaptation module can imitates the one learned by the expert. With Regularized Online Adaptation, the expert learns to predict environment extrinsics with regularization from the adaptation module, thus tiny imitation error, **resulting in 20% reduction in EE Error**. Table 2.6 shows that adding regularization to expert has negligible negative impact on performance, while every metric gets improved compared to RMA due to smaller latent imitation error. Note that DR has better base acceleration and total energy as it just stands in place under difficult environments.

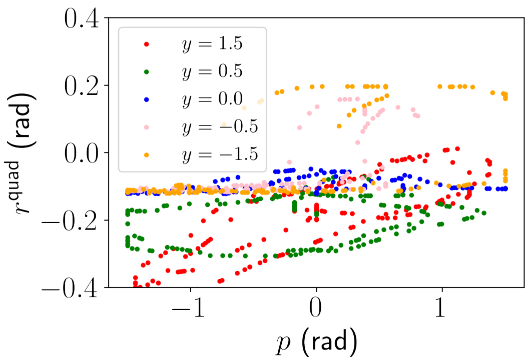
2.3.3 Real-World Experiments

We use the built-in Go1 MPC controller and the IK solver for operational space control of WidowX as the baseline in the real world, which we refer to as MPC+IK. More details are in the Supplementary.

Teleoperation: We specify EE position command p_t^{cmd} by parameterizing $p_{t+1}^{\text{cmd}} = p_t^{\text{cmd}} + \Delta p$, where $\Delta p = (\Delta l, \Delta p, \Delta y)$ is specified by two joysticks. With human in the loop, we can command the end-effector to reach points within or outside of training distribution. In Figure 2.5, we analyze the whole-body control in the real world, and show that the quadruped's base rotation $(r^{\text{quad}}, p^{\text{quad}}, y^{\text{quad}})$ strongly correlates



(a) Body pitch p^{quad} correlates with command EE pitch p with various lengths l.



(b) Body pitch r^{quad} correlates with command EE pitch p with various yaws y.

Figure 2.5: Real-world whole-body control analysis. (a) We fix command EE yaw $y = 0$ and change command EE pitch p and length l. When p has a large magnitude, the quadruped will pitch upward or downward to help the arm reach for its goal. With larger l (goal far away), the quadruped will pitch more to help. (b) When the magnitude of command EE yaw y is closer to 1.578 (arm turns to a side of the torso), the quadruped will roll more to help the arm. When $y = 0$, the quadruped pitches downward instead of roll sideways to help the arm.

with the EE position command p_t^{cmd}. This indicates that our unified policy enables whole-body coordination where the leg joints, as well as the arm joints, help reaching.

Vision-Guided Tracking: In addition to joystick control by humans, we also show successful picking tasks using visual feedback from an RGB camera. We mount a Realsense D435i camera near the gripper of the arm and use AprilTag [62] to get the relative position between the gripper and the object to be picked up. AprilTag is a visual fiducial system popular in robotics research using simple 2D black and white blocks to encode pose information. We first get the translation of the AprilTag in the camera frame $p^{\text{tag}} = [x^{\text{tag}}, y^{\text{tag}}, z^{\text{tag}}]^T$. Then we design and use a simple yet effective position feedback controller to set the current EE position command $p_t^{\text{cmd}} = K^T p^{\text{tag}}$, where $K = [-1.5, -1.5, 0.1]^T$ is a gain vector for position control. In Figure 2.6 and Table 2.7, we compare our method and the baseline (MPC+IK) in several pick-up tasks by measuring the success rate, average time to to completion (TTC), IK failure rate, and self-collision rate for every setting. We initialize the robot to the same

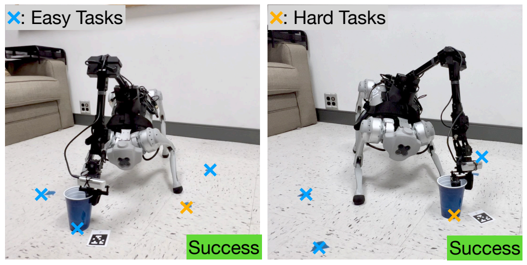
	Success Rate ↑	TTC ↓	IK Failure Rate ↓	Self-Collision Rate ↓
Easy tasks (tested on 3 points)				
Ours	**0.8**	**5s**	-	**0**
MPC+IK	0.3	17s	0.4	0.3
Hard tasks (tested on 5 points)				
Ours	**0.8**	**5.6s**	-	**0**
MPC+IK	0.1	22.0s	0.2	0.5

Table 2.7: Comparison of our method v.s. MPC+IK on pick-up tasks. p^{end} is the goal position sampled from the points on the ground. TTC is the average time to completion. Each task performance is averaged on 10 real-world trials.

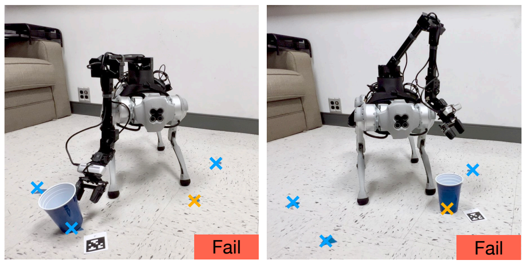
default configuration and before execution.

Analysis of Success and Failure Modes: Our method succeeds most of times on easy tasks without visible performance drop in hard task. The failed trials of our method are largely due to the mismatch between the actual cup position and the AprilTag position, which can be mitigated by using two AprilTags and averaging their poses (details in the Supplementary). Since the visual estimation is not the focus of this work, we infer that our method has higher precision and higher efficiency on pick-up tasks than MPC+IK. MPC+IK succeeds in some of the easy tasks and fails due to IK singularity or self-collision. In hard tasks, the major failure cause is self-collision given the cup is too close to the body. Notice that the TTC of MPC+IK is also longer than our method because solving online IK and operational space control more computationally demanding than joint position control (ours).

Open-loop Control from Demonstration: In this part, we analyze how agile walking is coupled with dynamic arm movement. The robot is given a pre-defined end-effector trajectory to follow in an open-loop manner while being commanded to walk at the same time. Results in Figure 2.7 show **agility** and **dynamic coordination** of our legged manipulator on uneven grass terrain powered by our whole-body control method.



(a) Our method: success in both easy and hard tasks with coordinated behaviors.



(b) MPC+IK: failure in both scenarios. Left: fail due to missed cup. Right: fail due to self-collision.

Figure 2.6: Comparison of our method and the baseline controller (MPC+IK) in vision-guided pick-up tasks. We sample different points around the robot as the target pick-up position. Easy tasks: 3 points are in normal distance from the robot. Hard tasks: when the point is very close to the front feet and are hard to reach without whole-body control. More hard tasks are in the Supplementary. Videos are at `https://maniploco.github.io`

2.4 Related Work

Legged Locomotion Traditional model-based control methods for legged robots have shown success but often require controllers to be meticulously designed and many manual tunings [5, 6, 10, 24, 27, 29, 30, 34, 36, 59, 70, 79, 89]. The extra weight and movement of a robot arm on top of the legged robot will make such design process more challenging. Recent advances in reinforcement learning enable legged robots to traverse challenging terrains and adapt to changing dynamics [14, 21, 22, 28, 39, 43, 48, 64, 77, 78, 87, 90, 91, 92, 93, 94]. However these works only focus on the mobility part and few interactions with objects or the environment by manipulation are studied.

Mobile Manipulation Adding mobility to manipulation is studied in [3, 8, 11, 16, 17, 26, 51, 80, 84, 85, 95]. Advances have also been made in the field of biped humanoid [13, 19, 37, 69]. More recently, Ma et al. [51] proposed using an MPC controller to track the desired end-effector position of the arm mounted on a quadruped with a RL policy to maintain balance. However, the controllers for legs (RL) and arm (model-based) are separate modules and no dynamic movements are demonstrated. In [3], language models are used to guide a mobile robot to finish different tasks

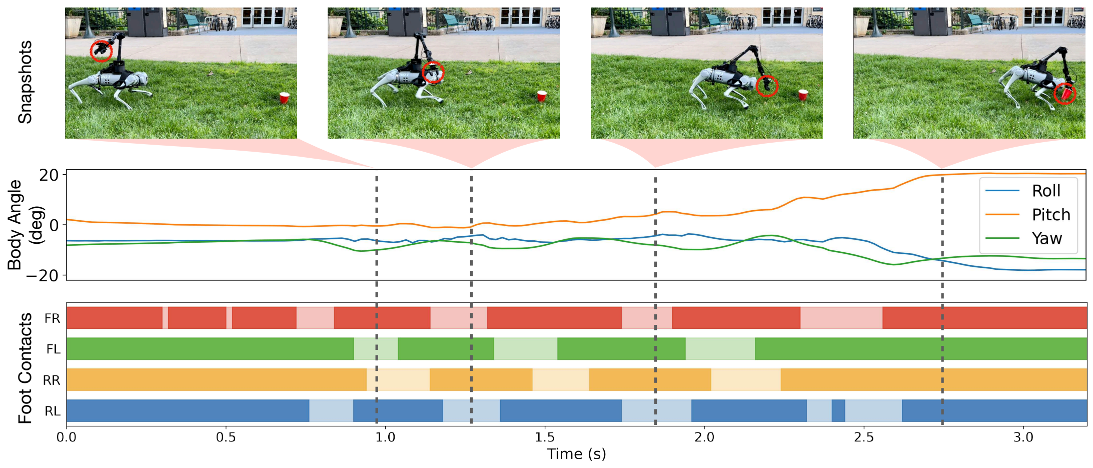


Figure 2.7: The arm follows a demonstration trajectory to pick up a cup while walking. The start position is $p = (0.5, -0.5, -1.2)$, at the right upper side of the robot and the end position is $p^{\text{end}} = (0.55, -0.9, 0.4)$, on the left lower front side close to the ground. $T_{\text{traj}} = 2.5s$. The robot initially stands on the ground and then is commanded by a constant forward velocity $v_x^{\text{cmd}} = 0.35$. Meanwhile the EE position command changes. When EE position command is high, the quadruped starts to walk without significant tilting behavior with a natural walking gait. As the EE position command moves below the torso, the quadruped starts to pitch downwards, roll to the left and yaw slightly to the right to help the arm reach the goal.

using the arm. However, the manipulation and mobility are utilized in a decoupled step-by-step manner.

2.5 Discussion and Limitations

We proposed a hardware setup as well as an algorithm to learn whole-body control of a legged robot with robotic arm. Our policy shows coordination between legs and arm while being able to control them in a dynamic manner. Although we have shown preliminary results on object interaction (e.g. picking, pressing, erasing), incorporating general-purpose object interaction (e.g. occlusion and soft object) into the our unified policy is a challenging open research direction. There are several ways in which the current methodology could be extended, such as, learning vision-based policies from the egocentric camera mounted on torso [2] and on the arm, climbing on the obstacle using front legs to pick something up on the table where the arm alone cannot reach, and etc. We believe this paper provides a first step towards several of such future directions.

Chapter 3

Legs as Manipulator: Pushing Quadrupedal Agility Beyond Locomotion

3.1 Introduction

Robotic quadrupeds and bipeds can walk across challenging scenarios ranging from hiking on hills to walking over rocky surfaces near river beds [2, 21, 31, 38, 47, 49, 54, 58, 63, 71, 88]. What is next for legged systems? Despite being highly effective walkers, legged robots are far behind the dexterity and agility of quadrupeds in the animal kingdom, such as dogs or cats, who can use their legs more generally than just for the task of walking. In particular, quadrupeds sometimes use their legs to open a door, dig a hole, pull an object, etc. [68]. Indeed, locomotion and manipulation can be seen as dual of each other [33, 50] and share evolutionary origins where frontal legs evolved to become arms in bipeds. Providing robotic quadrupeds a similar ability would not only push the *agility* of robotic locomotion but can also greatly expand the reach as well as functionality even if there is an arm already attached on top of the quadruped.

In this work, we focus on this joint problem of learning locomotion as well as basic manipulation skills by expanding the capability of quadruped robots to enable

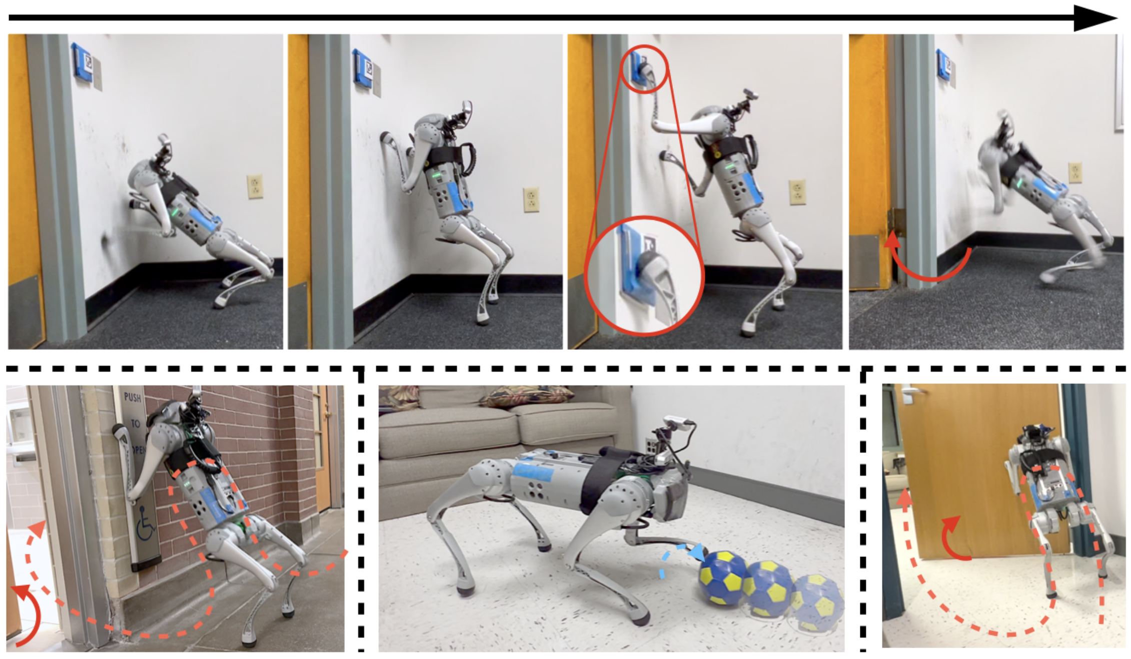


Figure 3.1: Examples of real-world skills. Top row: Robot climbs high on the wall to operate a wheelchair access button using its leg and then get off the wall to walk out of the door. Bottom left: Robot climbs on the wall and uses its weight to press the button. Bottom middle: Robot kicks a ball on the ground. Bottom right: Robot climbs a door and opens it using its weight and then walk indoors.

them to use their legs as manipulators. In particular, as shown in Fig. 3.1, we focus on tasks like climbing a wall with front legs, jumping on a wall to reach a button, using a leg to push a button, etc; and then combining them to achieve long-range behaviors. More broadly, we follow the popular approach of learning environment latent conditioned policies in simulation using reinforcement learning (RL) and then transferring them to the real world via sim2real [38, 42].

A major challenge in this technique for training legs to simultaneously walk and perform skills like climbing is that the RL can get stuck in local minima. To get around this issue, we decouple the skill policy training into *locomotion* and *manipulation* using legs. Locomotion policy captures anything that involves movement such as walking or climbing onto a wall and is conditioned on the commanded linear as well as angular velocity commands. On the other hand, the manipulation policy captures moving a single leg for tasks like pressing the button using one foot while balancing

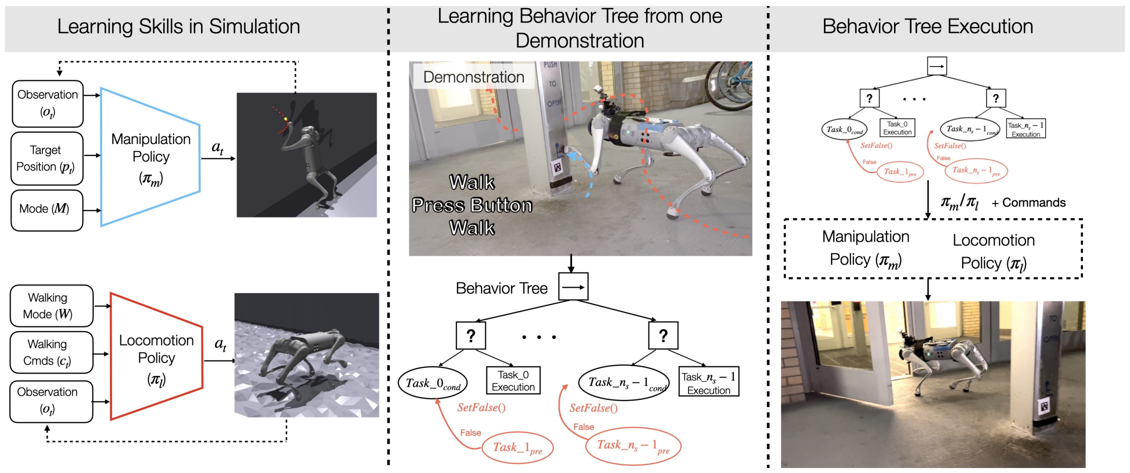


Figure 3.2: Overview of the method. Left: Skill learning framework where the two policies are trained separately with their own observations o_t^l and o_t^m in simulation. The user commands include boolean values to control different modes of walking (walk vs climb), to activate button pressing, and velocity commands. Middle: Learning a behavior tree from a single high-level demonstration which is primarily responsible for selecting between the walking and manipulating policies, as well as the mode to activate. Right: Once the behavior tree is learned, we execute the behavior tree under external interruptions and perturbations to show robust execution. The $\longrightarrow$ represents a Sequence node that executes its children in left-to-right order until all children return success. The ? is a Selector node that returns success or running whichever it encounters first in its children. The oval is a condition node that returns success when the condition is met and failure otherwise. The rectangle is the execution node that always return running. Videos at https://robot-skills.github.io/.

on other legs. This button pressing can be performed on a horizontal ground or a vertical wall.

We train these skills in simulation and transfer them to the real world. However, skills like climbing are highly agile and diverse in their behavior, thus relying on just onboard sensors is not sufficient. To stabilize the training, we need additional state estimates (velocity and foot contacts) which may not be directly available via a sensor. To resolve this, we follow an approach similar to RMA [38] to get an estimate of a unified state estimator (USE) via proprioception history, which can then be reliably computed onboard. Once we have policies that work in simulation, we use our proposed variant of regularized online adaptation (ROA) [23] to achieve a higher performance than off-the-shelf methods such as RMA [38].

Once we have a quadruped that can perform diverse skills, it must also figure out how to stitch these skills together to perform long-term planning tasks in a manner that is robust to the failures in underlying skills [65, 66, 67]. We propose to leverage the classic idea of behavior trees [7, 15] and adapt it to our learning paradigm. In particular, after training low-level skills, we learn a behavior tree from a single clean demonstration of a long-range task and show a robust replay of the behavior despite interruptions by introducing visually matched preconditions, recording the success of a low-level skill on completion, and a recursive backing up behavior under unexpected interruptions and perturbations.

We show extensive evaluation in both simulation and the real world. We use the learned skills to perform a variety of real-world tasks where robot climbs and presses buttons on walls or the ground.

3.2 Related Work

Many works study locomotion and some works study using legs for manipulation. However, few works have truly investigated how to combine locomotion and manipulation skills for real-world tasks. We review these below.

Legged Locomotion

Recent advances in reinforcement learning have enabled a set of works that train a neural network based walking controller for legged robots [2, 21, 31, 38, 47, 58, 63, 71]. How to achieve various gait patterns for quadrupedal robots that are emergent on real dogs is studied in [21, 54, 88]. [47, 54] train gait parameters conditioned policies to get diverse walking behavior that can achieve simple object manipulation such as pitching down to unload a ball on the back of the robot. However, these emergent individual skills are not dexterous enough and have not been synthesized to accomplish more complex tasks.

Legged Manipulation

Manipulation via locomotion has been studied in [16, 32, 60, 76, 86]. [76] proposed to use all the four legs of a quadrupedal robot as a dexterous manipulator by lying the robot back on the ground. However this scarifies the mobility of the legged system and makes it no difference from a four finger manipulator. [32] enables the quadruped to kick a ball to a goal with real-world finetuning but is still a single skill result without any agility in locomotion and lacks generalization to long-horizon tasks.

Skill Synthesizing

The ability to synthesize a vast variety of motor skills is important to achieve long-horizon complex behaviors for robots. Prior works [65, 66] combine adversarial imitation learning and unsupervised reinforcement learning to develop skill embeddings, and then train a high level policy to synthesize these low-level skill conditioned on task-specific rewards. [44, 45] aim to address the problem of transitions between different skills to chain long-horizon tasks whose success rate can be exponentially reducing due to transition failures.

Behavior Trees

Long-horizon tasks can be expressed with various forms of abstractions including decision trees, finite state machines or a neural network. Behavior trees are one of these forms that are firstly vastly used in game industry for non-player character control. With its modularity, transparency and execution speed, they are starting to be used more in robotics [7, 15, 20, 57]. The modularity helps the user to modify the behaviors without affecting other parts. The transparency gives more ease understanding the behaviors. And execution speed ensures reactive behaviors encountered with unknown environment changes.

3.3 Method: Legs as Manipulators

Our proposed skill learning and composition framework is shown in Fig. 3.2. We first learn low level skills in simulation and transfer them to the real world via online adaptation. Once we have these skills available in the real world, we then use behavior trees to learn to compose these skills from a single clean demonstration, and show robust replay in the real world despite interruptions and perturbations. We will now describe each component in detail.

3.3.1 Learning Low-level Skills

Walking and manipulation using legs include drastically different joint angle behaviors. Hence, for training stability, we decouple the problem into locomotion policy π_l to walk, and a policy to manipulate with legs π_m. Locomotion skill deals with movements, i.e. both walking and climbing, while manipulation involves using one leg to interact while using others to balance. Inspired by RMA [38], we want to learn adaptive policies which use an online estimate of environment to adapt their behaviors appropriately. Such polices can walk in the real world under a diverse set of conditions. Concretely, each policy takes the current state o_t (roll, pitch, base angular velocities, velocity commands, joint positions and velocities, base velocity, feet contact indicators), the extrinsics vector z_t (which is an estimate of the environment parameters), and additional task specific inputs. Then each policy outputs the target joint angles $a_t \in \mathbb{R}^{12}$ at 50Hz. We train these policies in simulation using model free reinforcement learning [72]. We will now describe the RL training setup which includes the additional task specific inputs, as well as the task specific reward terms which are used in addition to the following reward terms shared between both the policies to encourage smooth and energy efficient motions:

- Joint Acceleration: $-||\ddot{q}_t||_2$
- Action Rate: $-||a_t - a_{t-1}||_2$
- Hip Position: $-||q_{hip}||_2$
- Work: $-|\tau_t^T \cdot \dot{q}_t|$
- Z Velocity: $-||v_z||_2$

where v_z is vertical linear velocity, $\dot{q}_t, \ddot{q}_t, \tau_t$ are the joint velocities, accelerations and torques, and q_{hip} is the position of the hip motor. These reward terms are respectively weighted with 2.5e-7, 5e-3, 0.1, 3e-3, 1.

Locomotion Policy The goal of the locomotion policy π_l is to follow a target command velocity (v_x^{cmd}, v_y^{cmd}, $\dot{\omega}_{yaw}^{cmd}$). The terrain in simulation is half fractal terrains, and the other half is slopes of different incline θ_{slope}. The robot is initialized in the plane and its task is to walk closer to the wall and then climb it. Once it climbs the wall stably, it is commanded to de-climb the wall. To avoid local minima, we introduce *Terrain Curriculum (TC)* where the incline of the slope is gradually increased during the course of the training all the way to a vertical wall.

In addition to o_t and z_t described above, the robot observes, d_w, the closest distance from robot's base to wall edge (where the terrain starts to incline) clipped in $[0.25, 0.65]$m. We add the following task specific rewards:

- Linear Velocity Tracking: $exp(-5||v_{xy} - v_{xy}^{cmd}||_2)$
- Angular Velocity Tracking: $exp(-5||\dot{\omega}_{yaw} - \dot{\omega}_{yaw}^{cmd}||_2)$
- Goal Reaching: 1 if $d_l > d_{thresh}$ & $d_r > d_{thresh}$ else 0
- Feet Air : $\sum_{f=0}^{4} (\mathbf{t}_{air,f} - 0.5)$
- Slope State Regularization: $-||q_t - q_{manip}||_2$ if robot is holding on the wall

where $v_{xy} = [v_x, v_y]^T$ are the base linear velocities, $v_{xy}^{cmd} = [v_x^{cmd}, v_y^{cmd}]^T$ is the commanded velocity, $\dot{\omega}_{yaw}$ is the yaw rate and $\dot{\omega}_{yaw}^{cmd}$ is the commanded yaw velocity, $\mathbf{t}_{air,f}$ is the time in air for foot f. q_t is the joint position. d_l and d_r are the distances from the front left and right foot to the wall. $d_{thresh} = 0.7$ and q_{manip} is the resting pose on the wall from which π_m will get initialized.The weight values for each of the reward terms are: 1.5, 0.5, 1.5, 1.0 and 0.3 respectively.

Manipulation Policy Manipulation policy's goal is to follow a desired end-effector position $p_{foot}^{cmd}(t) = [p_x(t), p_y(t), p_z(t)]^T$ such that the foot can track any arbitrary pre-planned trajectory, even while resting on an inclined wall. For simplicity, we formulate all the trajectories as a sinusoidal trajectory in $[0, \pi]$ resulting in a behavior that first lifts the foot and then drop it. By specifying a start point $p_{foot}^{cmd}(0) = [p_x(0), p_y(0), p_z(0)]^T$ and an end point $p_{foot}^{cmd}(T) = [p_x(T), p_y(T), p_z(T)]^T$, we can interpolate a sinusoidal trajectory, parameterized by foot lift height H and duration T.

$$p_{foot}^{cmd}(t) = [s\Delta x + p_x(0), s\Delta y + p_y(0), H\sin(\pi s)] \tag{3.1}$$
$$s = \frac{\min(t, T)}{T}, \ \Delta x = p_x(T) - p_x(0), \ \Delta y = p_y(T) - p_y(0)$$

where t goes from 0 to T. Regardless of slope angle θ_{slope}, we always want the foot lift direction to be perpendicular to the slope plane. So $p_{foot}^{cmd}(0)$ and $p_{foot}^{cmd}(T)$ are all defined in the coordination frame whose origin is on the wall edge and xy axes are on the slope plane. Along with o_t and z_t described above, the policy additionally takes the timer variable s, foot lift height H, end point position viewed from robot's base frame $p_{foot_{base}}^{cmd}(T)$ as inputs. We add following rewards:

- Trajectory Tracking: $exp(-10(p_{foot}^{cmd}(t) - p_{foot}(t))$
- Slope States Regularization: $-||q_t - q_{manip}||_2$

where $p_{foot}(t)$ is the foot position in the same frame mentioned above. The weights for the terms are 2.0, 0.5. The sampling ranges for low-level commands are in Tab. 3.1.

3.3.2 Simulation to Real Transfer

We do not have access to the latent extrinsics z_t in the real world to deploy the above policies. To resolve this, we follow [38] and learn to estimate z_t from proprioception history. This is done by the adaptation module which we can train in simulation itself using supervised learning since we have access to ground truth values. However, we find that the adaptation module might not be able to faithfully reconstruct z_t, leading to an inferior overall performance. One way to recover the performance is to add a third phase by finetuning the motor policy with the estimate $\hat{z}_t$ instead of ground truth z_t, as in A-RMA [40]. However, in this paper, we simplify the setup by reducing three phases into a joint single stage training which internally alternates between training a teacher policy with ground truth environment dynamics parameters (z_t) and an estimator of z_t from proprioception history, inspired from Regularized Online Adaptation (ROA) [23]. Specifically, the privileged information encoder μ encodes information about the environment e_t into a latent vector $z_t \in \mathbb{R}^{20}$ and then passes to the base policy (π_l or π_m). The adaptation module ϕ trained using supervised learning by $z_t = \mu(e_t)$, while the privileged info encoder μ is encouraged to be close to $\phi(x_{t-10}, a_{t-11}, \cdots, x_{t-1}, a_{t-2})$. The ranges of environment dynamics randomization is same as [38].

However, unlike [23], our robot needs to perform highly varying and agile tasks like climbing during which on-board sensing can be noisy. Hence, we need a way to estimate the values such as robot's velocity or foot contacts in an online manner itself. To facilitate this, we train an additional unified estimator ω to estimate unavailable robot states (such as velocity and foot position). The estimated robot states are the base linear velocity $v_{base} = [v_x, v_y, v_z]^T$ for locomotion policy π_l, and manipulation foot position $p_{foot} = [x_{foot}, y_{foot}, z_{foot}]^T$ in robot's base frame. Similar to [31], we learn an estimator of v_{base} and p_{foot} from proprioception history in simulation. We

Policy	Command Vars	Training Ranges	Test Ranges
Loco	v_x^{cmd} (m/s)	[-1.0, 1.0]	[-1.5, 1.5]
	v_y^{cmd} (m/s)	[-0.6, 0.6]	[-1.0, 1.0]
	$\dot{\omega}_{yaw}^{\text{cmd}}$ (rad/s)	[-0.5, 0.5]	[-0.6, 0.6]
Manip	Δx (m)	[-0.1, 0.3]	[-0.15, 0.3]
	Δy (m)	[-0.1, 0.15]	[-0.2, 0.2]
	H (m)	[0.06, 0.2]	[0.05, 0.22]
	T (s)	[0.5, 2.0]	[0.3, 3.0]
Common	θ_{slope} (rad)	[0, 1.57]	[0, 1.57]

Table 3.1: Uniform sampling ranges of low-level commands

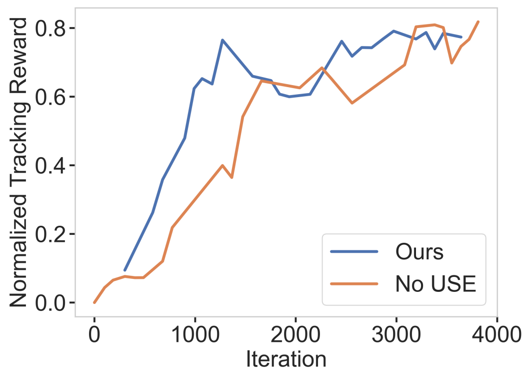
call this Unified State Estimator (USE). We show that USE, ROA, and TC, all are integral to achieving good performance.

3.3.3 Composing Skills into High-Level Behaviors

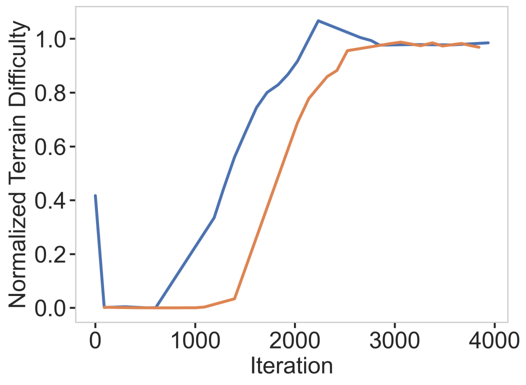
Finally, we combine the above learned skills into long-range behaviors from only one demonstration. Note this single long-range demonstration is only in the high-level action space where human chooses what skill to follow and when, the low-level control is taken care of by the skills (π_l, π_m) learned above. We distill this single demonstration into a behavior tree [15] to learn to robustly complete the task.

Behavior Tree Execution We assume our tasks are sequential and formulate our tree structure shown in Fig. 3.2. The behavior tree executes from the starting root node, sequentially checking the children's condition node to see which ones have succeeded, and executing the one yet to be completed. If all children of the root node returns success, then the entire task will be completed.

Behavior Tree Learning The human gives one demonstration of a long-term task. During the demonstration, we record the human commands and the corresponding visual representation vectors from the second to last layer of ResNet18 [25, 75] $v^r \in \mathbb{R}^{512}$. From the demonstrations, we create a tree whose number of children is



(a) Tracking performance



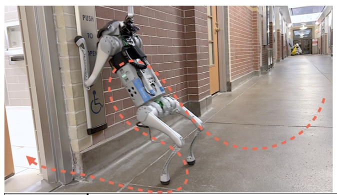
(b) Terrain difficulty

Figure 3.3: USE helps the locomotion policy to achieve better performance and also boosts the training speed. The Normalized Terrain Difficulty ntd indicates the maximum slope angle that the policy can climb θ_{slope}. $ntd = 1$ means θ_{slope} is uniformly sampled from the range in Tab. 3.1. USE accelerates the policy learning, showing faster convergence to final performance for linear and angular velocity tracking. USE also helps the policy to advance to more challenging terrains with a larger Slope Angle θ_{slope} faster.

equal to the number of skills executed n_s. The goal of child node $i \in [0, n_s - 1]$ is to execute the i^{th} high-level command until the corresponding condition node $Task\_i_{cond}$ returns true. The criteria for deciding $Task\_i_{cond}$ is a visual representation score in the form of cosine distance between current representation vector v_t^r and that of expert's $D_r[i]$ when its subskill is complete, $score_i = 1 - \frac{v_t^r \cdot D_r[i]}{||v_t^r||_2 \cdot ||D_r[i]||_2}$.

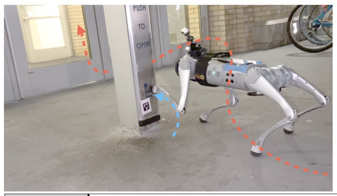
$$Task\_i_{cond} = \begin{cases} True, & score_i < 0.16 \\ False, & Otherwise \end{cases}$$

Robustness via Task Precondition Every task has a precondition $Task\_i_{pre} = f(C[i], o_t, p_t, c_t)$, which is a function of the current robot states. If $Task\_i_{pre} == True$ is not satisfied, we rewind to the previous task recursively until we find a proper task that satisfy its own precondition. The tree structure introduced above is robust to interruptions or perturbations by design. For example, if the robot slips right before pressing the button high up on the wall, then the precondition variable $Task\_i_{pre}$ of climbing on the wall will be $False$, and that will allow the behavior tree to backup, and re-execute the nodes which are incomplete.



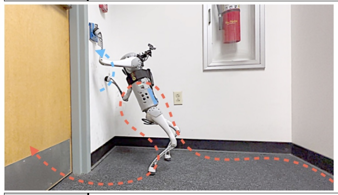
a)	Subskills Completed	Subskill
Ours	3/3	Walk Climb Walk
w/o RB	1.2/3	
Replay	1/3	

Goal	Use robot's body weight to press the button
Perturbation	Put an obstacle in front of the robot when the robot is about to climb
Precondition	Ensure no obstacle before invoking the climbing behavior



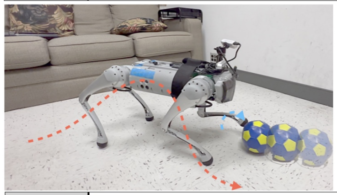
b)	Subskills Completed	Subskill
Ours	5/5	Walk Balance Press Balance Walk
w/o RB	2.6/5	
Replay	2.6/5	

Goal	Press the button near the ground to go inside the building
Perturbation	Pull the robot away from the button when it is about to switch to press it.
Precondition	Button should be in sight before activating the button press action.



c)	Subskills Completed	Subskill
Ours	6.2/7	Walk Climb Balance Press Balance Climb Walk
w/o RB	2.6/7	
Replay	1.8/7	

Goal	Press the button that high on the wall to go outside the building
Perturbation	Remove the robot off the wall before the button is pressed
Precondition	Robot should be on the wall before activating the button press action



d)	Subskills Completed	Subskill
Ours	5/5	Walk Balance Press Balance Walk
w/o RB	2.6/5	
Replay	2/5	

Goal	Kick a ball on the ground
Perturbation	Move the ball farther away when the robot is about to kick it
Precondition	The ball should be close before activating the kicking behavior

Walk: $\pi_m(W=0)$ Climb: $\pi_m(W=1)$ Balance: $\pi_l(M=0)$ Press: $\pi_l(M=1)$

Figure 3.4: Our method outperforms baseline methods and achieves the highest subskills completion rate. The red dashed line indicates the trajectory of the robot base. The blue line indicates the manipulation foot trajectory when π_m is in use. We list all the subskills at the bottom and also list the sequence of skills needed for each long-range task next to it. Subskills Completed measures the fraction of selector nodes marked as a success. The data is averaged during 5 real-world trials for each task.

3.4 Results, Setup and Analysis

We quantitatively analyze a) the benefit of USE, ROA, and TC in simulation, b) compare our method with baselines on long-range tasks with interruptions, and c) show the robustness of our method on a very long-range task in the real world to show how our method enables the quadruped to expand its reach in real-world environments.

Hardware and Simulation Details We use the Unitree Go1 quadrupedal robot and one forward-facing Realsense D435i camera to estimate distance to the wall and another side-facing one to detect an AprilTag[61] for button location. We use IsaacGym [53] as our simulation platform.

	Vel Error↓	Traj Error↓	z_t loss ↓ Loco	z_t loss ↓ Manip
Ours	**29** $\pm$ 11	**3.1** $\pm$ 3.0	8.6 $\pm$ 6.3	**6.5** $\pm$ 6.0
w/o USE	32 $\pm$ 19	3.7 $\pm$ 3.9	**7.2** $\pm$ 3.6	6.8 $\pm$ 7.1
RMA ([38])	39 $\pm$ 30	4.7 $\pm$ 6.1	92 $\pm$ 33	53 $\pm$ 27
w/o TC	58 $\pm$ 17	-	9.3 $\pm$ 5.5	-
DR	41 $\pm$ 31	5.8 $\pm$ 8.8	-	-
Ours (priv)	24 $\pm$ 9.0	3.2 $\pm$ 3.3	-	-
RMA (priv)	23 $\pm$ 6.8	3.1 $\pm$ 3.4	-	-

Table 3.2: Low-level commands tracking performance in simulation ($1e-2$). Our method outperforms all baselines in tracking error. The z_t loss is defined as $||\mu(e_t) - \phi(x_{t-10}, a_{t-11}, \cdots, x_{t-1}, a_{t-2})||_2$. We collect data equivalent to 10 hours real-world time. In Domain Rand, the robot does not observe environment extrinsics z_t.

3.4.1 Simulation Experiments

We show the quantitative importance of the Unified State Estimator (USE), Regularized Online Adaptation (ROA), and Terrain Curriculum (TC) in learning low-level skill execution in simulation across the following metrics: a) velocity tracking error, b) trajectory tracking error, c) z regression error. We also compare to *DR* (Domain Randomization), and Policies with ground truth extrinsics z_t (*Ours-priv* and *RMA-priv*). We add external pushes during evaluation (Tab 3.2).

Regularized Online Adaptation The regularization does not degrade the expert's performance in ROA, while still showing a significant improvement in the student policy (about **30**% for π_l and π_m) compared to RMA [38], achieving an order of magnitude improvement in regressing to z_t.

Unified State Estimator To demonstrate the effectiveness of using USE in locomotion policy, we compare how USE improves policy learning during training as shown in Fig. 3.3. With USE the policy reaches similar performance 10% faster than the baseline method which does not employ USE.

Terrain Curriculum Without TC, the velocity error is **107**% larger than our method since the policy simply learns to walk until approaching the wall and stop.

With TC, the robot can learn to climb the relatively challenging vertical wall by first learning to climb a slope with a lesser incline.

3.4.2 Evaluating Robust Skill Composition in Real World

We first compare our method with baselines on long-range tasks with interruptions to test the robustness in the real world. We define 4 setups that require the use of legs for manipulation (Fig 3.4) with the number of subskills $n_s \in \{3, 5, 7\}$. For each of these settings, we learn the behavior tree from one clean demonstration and evaluate the performance of our method with the stated interruption. Since we only have one interruption per task, we simplify our experiment design and define only one precondition corresponding to the subskill affected by the interruption. This precondition can be used to determine if the behavior tree needs to rollback.

Fig. 3.4 compares our method's performance to baselines:

- w/o RB: Use behavior tree to specify high-level commands but without task precondition. This version runs the nodes in sequence, switching only when the current task is completed, but fails to roll back to redo a previously finished task in case of an interruption.
- Replay: Record the expert's high-level commands with time stamps, and simply replays them for the same amount of time as in the demonstration.

We measure the subskill completion rate, as well as the overall task completion rate for each of the tasks across 5 trials (Fig. 3.4). We can see that our method achieves the highest subskills completion rate across trials. While w/o RB can complete subskills until the interruption, it is unable to complete the entire task. In a), w/o RB can sometimes climb the wall with the box obstacle under its body. In comparison, our method can switch back to walking mode and find another spot without an obstacle to complete the task. In b), w/o RB stretches its leg to the limit and manages to press the button if the robot is not moved too far away, while our method will switch back to walking when the button is too far. In c), w/o RB will never climb the wall again when it is removed off the wall while our method falls back to climbing again and then completing the task. Replay performs the worst since it uses no feedback on task completion. It naively executes the commands with the same timestamps as recorded during the expert demonstration.

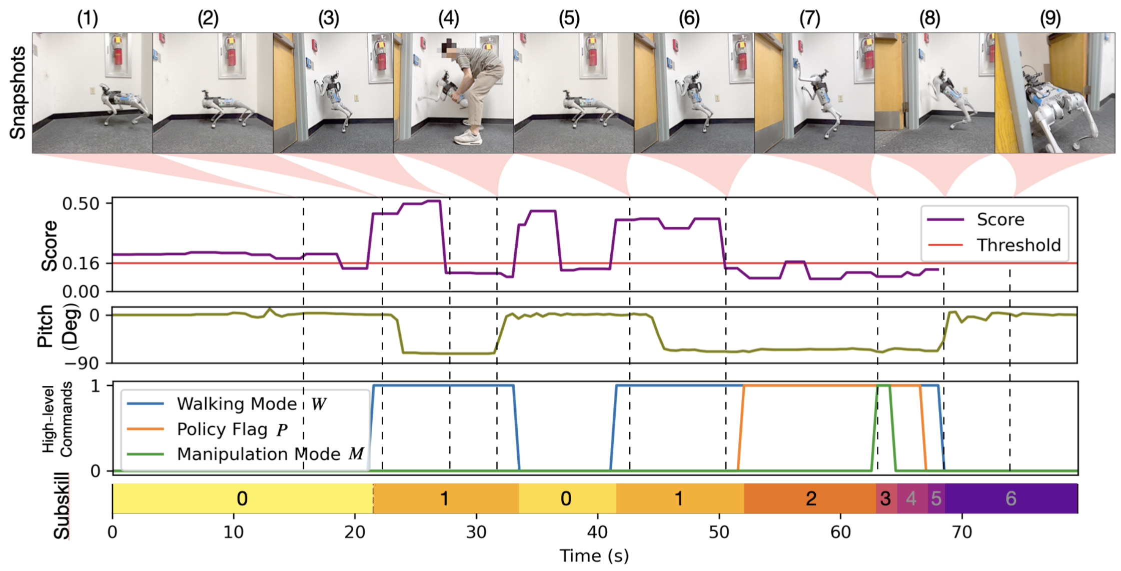


Figure 3.5: Demonstration of a long horizon door-opening task with learned behavior tree from expert demonstration. The subskill tuples are specified in bottom left of Fig. 3.4. The score is the cosine distance between the current visual representation vector and that of the expert's when the expert activates that subskill. Pitch is the pitch angle of the robot base. Subskill indicates which skill in the behavior tree is currently executing. Note that for subskill 6 there is no score because that is the last one.

3.4.3 Analysis on a Long-Horizon Task in the Real World

We now show a detailed analysis of how our framework works on a task that involves 7 subskills and 6 skill switches as shown in Fig. 3.5. This task shows the expanded capability enabled by our low-level skills as well as the robustness of skill composition. The robot is able to press a wheelchair access button that is 0.95m above the ground and walk out of a door. This task itself requires the robot to utilize two rear legs to raise the base (which is otherwise only 0.5m high), and then use its legs for interaction. It stretches the left front leg to reach the otherwise impossible-to-reach button and then presses it. The button is located in a corner surrounded by walls on three sides. The space is about 2m long and 0.5m wide, which tight relative to the robot's dimensions (0.5m long and 0.3m wide). The accurate low-level command tracking performance makes it precise enough to enable movement in this tight space.

In detail, the robot first needs to walk from an open field to this narrow corner as shown in Snapshot (1). This is recorded by the behavior tree as a success after verifying that the visual representation score is smaller than the threshold. The robot then switches to climbing and climbs onto the wall ((2), (3)). At this time, we manually perturb the robot and remove it off the wall and on the flat ground. Without task precondition, the behavior tree continues to try to press the button, although it first needs to climb the wall again. This leads to unpredictable and unsafe joint movements since it has never been trained in this scenario. However, with task precondition (*ours*), the removal from the wall violates the precondition for button pressing, since the pitch angle of the robot exceeds -1. The behavior tree goes back to the previous task that satisfies its own precondition ((4)), and re-runs through the tree from that node. Consequently, the robot starts over to climb the wall ((5), (6)), presses the button ((7)), and gets off the wall to go out the opened door ((8), (9)). The system can handle multiple external perturbations, although, we only show 1 successful recovery here.

3.5 Conclusion

In this paper, we present a framework for synthesizing low-level skills for quadrupedal robots that involve locomotion and manipulation using legs. We then compose these skills to achieve long-range complicated tasks that gives robots more capability to access human environments. Our framework is both agile and robust and aims to push the limits of robotic quadrupeds. Our robot performs a set of useful real-world tasks including opening doors by pressing various types of buttons of different heights and interacting with moving objects. Rigorous evaluations are performed both in simulation as well as real world scenarios. A limitation is that we currently decouple high-level decision making and low-level command tracking and making it fully end-to-end is an exciting future direction.

Appendix A

Appendix for Deep-Whole Body Control

A.1 Experiment Videos

We perform thorough real-world analysis of our framework and our custom-built legged manipulator. We urge the reader to look at the compiled result videos at `https://maniploco.github.io` . As we can see in the video, legs and arm function in coordination with each other where legs bend and stretch to increase the reach of the arm as well as to attain stability.

A.2 Regularized Online Adaptation Details

We presented the details of Regularized Online Adaptation (Section 2.2 of the main paper) in Algorithm 1. We set H to be 20. The regularization coefficient λ follows a linear curriculum which starts at 0 and stops at 1: $\lambda = \min(\max(\frac{itr-5000}{5000}, 0), 1)$.

A.3 Simulation Details

We obtained URDF files for the quadraped and the robot arm from Unitree and Interbotix separately. We customized the URDF files to connect the two parts rigidly.

Algorithm 1 Regularized Online Adaptation

```
 1: Randomly initialize privileged information encoder $\mu$, adaptation module $\phi$,
    unified policy $\pi$
 2: Initialize with empty replay buffer $D$
 3: for $itr = 1, 2, \dots$ do
 4:     for $i = 1, 2, \dots, N_{env}$ do
 5:         $s_0, e_0 \leftarrow$ envs[i].reset()
 6:         for $t = 0, 1, \dots, T$ do
 7:             if $itr$ mod $H == 0$ then
 8:                 $z_t^{\phi} \leftarrow \phi(s_{t-10:t-1}, a_{t-11:t-2})$
 9:                 $a_t \leftarrow \pi((s_t, a_{t-1}, z_t^{\phi}))$
10:             else
11:                 $z_t^{\mu} \leftarrow \mu(e_t)$
12:                 $a_t \leftarrow \pi((s_t, a_{t-1}, z_t^{\mu}))$
13:             end if
14:             $s_{t+1}, r_t \leftarrow$ envs[i].step($a_t$)
15:             Store $((s_t, e_t), a_t, r_t, (s_{t+1}, e_{t+1}), z_t^{\phi}, z_t^{\mu})$ in $D$
16:         end for
17:     end for
18:     if $itr$ mod $H == 0$ then
19:         Update $\theta_{\phi}$ by optimizing $||\text{sg}[z_t^{\mu}] - z_t^{\phi}||_2$
20:     else
21:         Update $\theta_{\pi}, \theta_{\mu}$ by optimizing $-J(\theta_{\pi}, \theta_{\mu}) + \lambda||z_t^{\mu} - \text{sg}[z_t^{\phi}]||_2$, where $J(\theta_{\pi}, \theta_{\mu})$
    is the
22:         advantage mixing RL objective in Section 2.1 of the main paper
23:     end if
24:     Empty $D$
25:     $\lambda \leftarrow$ Linear_Curriculum($itr$)
26: end for
```

Shown in Figure A.1, we use Nvidia's IsaacGym [52] for parallel simulation. We

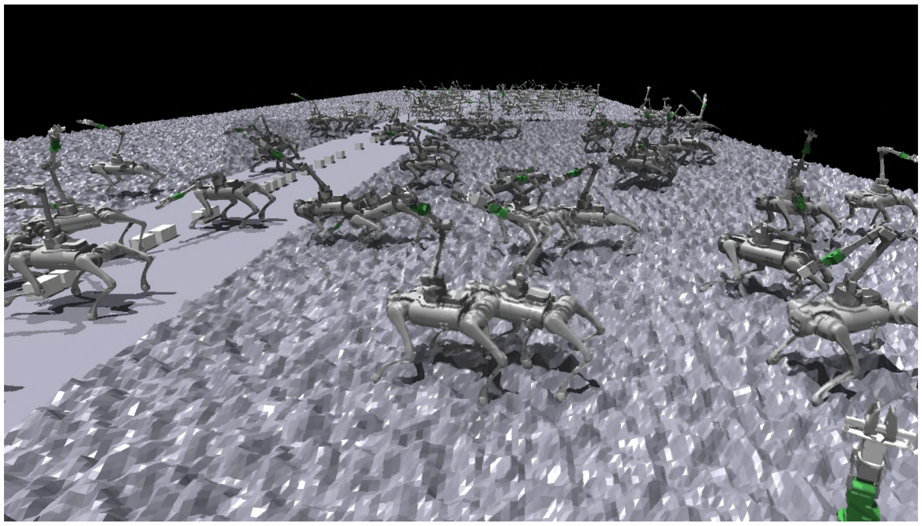


Figure A.1: Customized simulation environment based on IsaacGym

use fractal noise to generate the terrain. The parameters for the fractual noise are number of octaves = 2, fractal lacunarity = 2.0, fractal gain = 0.25, frequency = 10Hz, amplitude = 0.15m. We found that the generated rough terrain will enforce foot clearance and replace the complex rewards that are needed if flat terrain is used for simulation [28].

We sample an EE position command by first sampling a spherical coordinate (l, p, y) from Table 2 of the main paper. Then world coordinate of p^{end} is obtained as $T(\text{S2C}[(l, p, y)]) + (p_x^{\text{base}}, p_y^{\text{base}}, p_z^{\text{base}})$, where T is the linear transformation according to the base orientation, S2C[] is the operator to transform spherical coordinates to Cartesian coordinates, and p^{base} is the base position. To encourage smooth arm motion and whole-body coordination, we set p_z^{base} to be a constant (0.53) and row and pitch in T to be 0, so EE position commands are z, row, pitch-independent of the base.

We simulate each episode for a maximum of 1000 steps and terminate the episode earlier if the height of the robot drops below 0.28m, body roll angle exceeds 0.2 radians if EE position command if on the left of the body base ($p_{\text{y}}^{\text{cmd}} > 0$) , or is less than -0.2 radians if EE position command is on the right of the body base ($p_{\text{y}}^{\text{cmd}} < 0$), or the body pitch exceeds 0.2 radians if EE position command is above body base ($p_{\mathrm{p}}^{\mathrm{cmd}} > 0$), or is less than -0.2 radians if EE position command is below body base ($p_{\mathrm{p}}^{\mathrm{cmd}} < 0$). We do not early terminate if the arm self-collide and any body parts with the terrain, but the EE command positions are sampled in a way that

The control frequency of the policy is 50Hz, and the simulation frequency is 200Hz. We set the stiffness (K_p) for leg joints and arm joints to be 50 and 5 respectively and the damping (K_d) to be 1 and 0.5 respectively. The default target joint positions for leg joints are $[-0.1, 0.8, -1.5, 0.1, 0.8, -1.5, -0.1, 0.8, -1.5, 0.1, 0.8, -1.5]$ and for arm joints are zeros. The delta range of target joint positions for leg joints is 0.45 and for arm joints are $[2.1, 1.0, 1.0, 2.1, 1.7, 2.1]$.

A.4 Training Details

The policy is a multi-layer perceptron which takes in the current state $s_t \in \mathbb{R}^{75}$, which is concatenated with the environment extrinsics $z_t \in \mathbb{R}^{20}$. The first hidden layer has 128 dimensions and after that the network splits into 2 heads, where each has 2 hidden layers of 128 dimensions. The outputs of two heads are concatenated, where the leg actions $a_t^{\mathrm{leg}} \in \mathbb{R}^{12}$ and arm actions $a_t^{\mathrm{arm}} \in \mathbb{R}^{6}$. We train for 10000 iterations / training batches, which are 2 billions of samples and 200k gradient updates. We list the hyperparameters of PPO [73] in Table A.1 of the Supplementary.

Table A.1: Training Hyper-parameters

PPO clip range	0.2
Learning rate	2e-4
Reward discount factor	0.99
GAE λ	0.95
Number of environments	5000
Number of environment steps per training batch	40
Learning epochs per training batch	5
Number of mini-batches per training batch	4
Minimum policy std	0.2

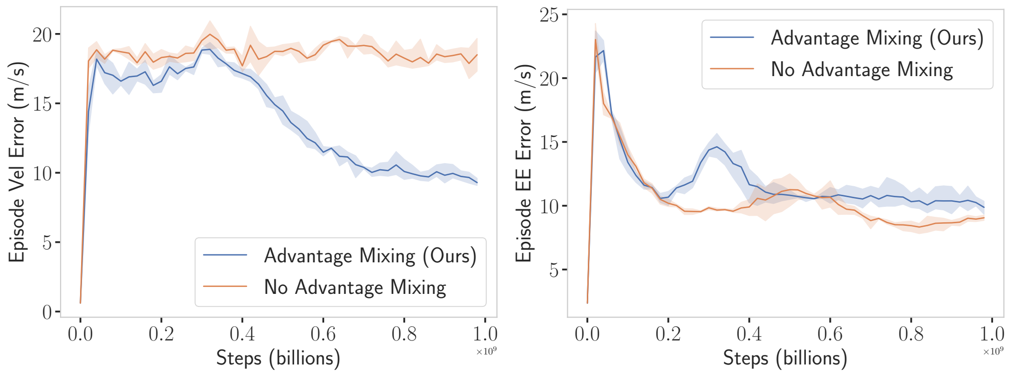


Figure A.2: Advantage mixing helps the unified policy to learn to walk and grasp at the same time. Without Advantage Mixing, the unified policy fails to learn to walk where the Episode Vel Error (episodic sum of L1 error between velocity commands and current velocities) is constantly high. In this case, the unified policy stays at local minima of only following EE commands.

A.5 Advantage Mixing Details

For a policy with diagonal Gaussian noise and a sampled transition batch D, the training objective with respect to policy's parameters θ_π is

$$\begin{aligned}
J(\theta_\pi) &= \frac{1}{|\mathcal{D}|} \sum_{(s_t,a_t)\in\mathcal{D}} \log \pi\left(a_t \mid s_t\right) A(s_t, a_t) \\
&= \frac{1}{|\mathcal{D}|} \sum_{(s_t,a_t)\in\mathcal{D}} \log\left(\pi(a_t^{\text{arm}} \mid s_t)\pi(a_t^{\text{leg}} \mid s_t)\right)\left(A^{\text{manip}}(s_t, a_t) + A^{\text{loco}}(s_t, a_t)\right) \\
&\to \frac{1}{|\mathcal{D}|} \sum_{(s_t,a_t)\in\mathcal{D}} \log \pi(a_t^{\text{arm}} \mid s_t)\left(A^{\text{manip}} + \beta A^{\text{loco}}\right) + \log \pi(a_t^{\text{leg}} \mid s_t)\left(\beta A^{\text{manip}} + A^{\text{loco}}\right)
\end{aligned}$$

In Figure A.2 of the Supplementary, we plot the episodic velocity command following error (Episode Vel Error) and EE comand following error (Episode EE Error) against number of steps during training. Advantage mixing helps the unified policy to learn to walk and grasp at the same time. Without Advantage Mixing, the unified policy fails to learn to walk where the Episode Vel Error (episodic sum of L1 error between velocity commands and current velocities) is constantly high. In this case, the unified policy stays at local minima of only following EE commands by not exploring in leg action space, since the initial exploration phase in leg action space will destabilize the base which harms manipulation tasks.

A.6 Real-World Setup and Experiment Details

Table A.2: Camera Parameters for Vision Tracking

Resolution	640 × 400
Frequency	10 Hz
Tag/Cam offset	(-0.02, -0.03, 0.12)

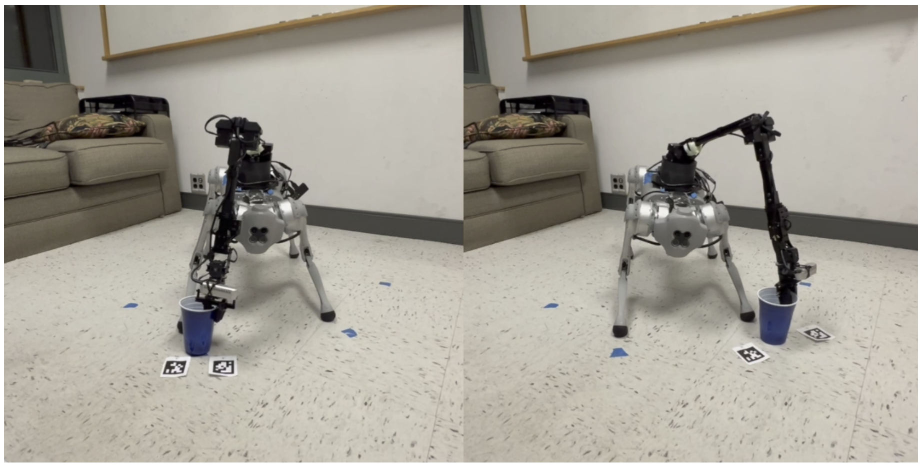


Figure A.3: Vision-guided tracking by using the average pose of the two AprilTags as the target pose.

The robot platform is comprised of a Unitree Go1 quadraped [82] with 12 actuatable DoFs, and a robot arm which is the 6-DoF Interbotix WidowX 250s [1] with a parallel gripper. We mount the arm on top of the quadruped. The RealSense D435 provides RGB visual information and is mounted close to the gripper of WidowX. Both power of Go1 and WidowX (60 Watts) are provided by Go1's battery.

In real-world experiments, we directly deploy the unified policy with the adaptation module with weights fixed onto the onboard computation of Go1, both modules operate at 50Hz. The inference of policy and adaption module are done on Raspberry Pi 4. The software stack of the WidowX 250s arm is setup on Nvidia TX2 by using the official codebase at https://github.com/Interbotix/interbotix_ros_

	Ground Points (p^{end})	Success Rate ↑	TTC ↓	IK Failure Rate ↓	Self-Collision Rate ↓
Easy tasks (tested on 3 points)					
Ours	$(0.62, -1.27, -1.11)$	**0.8**	**5s**	-	**0**
MPC+IK	$(0.57, -1.16, 0.55)$	0.3	17s	0.4	0.3
	$(0.58, -1.14, 1.78)$				
Hard tasks (tested on 5 point)					
Ours	$(0.72, -0.51, 0.34)$	**0.8**	**5.6s**	-	**0**
MPC+IK	$(0.55, -0.75, -0.43)$	0.1	22.0s	0.2	0.5
	$(0.56, -0.73, 0.5)$				
	$(0.45, -0.74, 1.80)$				
	$(0.45, -0.76, -1.8)$				

Table A.3: Comparison of our method v.s. MPC+IK on pick-up tasks. p^{end} is the goal position sampled from the points on the ground. TTC is the average time to tompletion. All data are averaged on 10 real-world trials.

`manipulators`. UDP is used as the communication protocal between Pi and TX2. EE gripper closing and opening are not a part of the policy.

In teleoperation experiments, the gripper action is directly controlled by a joystick controller. In vision-guided tracking experiments, we use a scripted policy to control the gripper: when the gripper position is close to the desired position specified by the AprilTag [62] for 1 second, the gripper closes; otherwise, it keeps open.

We listed the camera parameters used in vision-guided tracking in Table A.2 of the Supplementary. The "Tag/Cam offset" describes what the desired translation of the tag should be viewed in the camera frame when using the position controller to specify desired end-effector position in spherical coordinate. Shown in Figure A.3 of the Supplementary, we also performed additional experiments on vision-guided tracking suggested by Reviewer bkQw by using two AprilTags and averaging their pose to get the target pose. Video results are at here. We listed the positions of ground points for visual-guided tracking tasks in Table A.3. More results on hard tasks are at here.

Bibliography

[1] WidowX 250 robot arm 6DOF - X-Series robotic arm. https://www.trossenrobotics.com/widowx-250-robot-arm-6dof.aspx. 2.3.1, A.6

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning (CoRL)*, 2022. 2.5, 3.1, 3.2

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as I can, not as I say: Grounding language in robotic affordances. 2022. 2.4

[4] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 2.2.1

[5] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *IEEE Transactions on Automatic Control*, 2014. 2.4

[6] Monica Barragan, Nikolai Flowers, and Aaron M. Johnson. MiniRHex: A small, open-source, fully programmable walking hexapod. In *RSS Workshop*, 2018. 2.4

[7] Joseph Bates, A Bryan Loyall, and W Scott Reilly. Integrating reactivity, goals, and emotion in a broad agent. In *In Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*. Citeseer, 1992. 3.1, 3.2

[8] C Dario Bellicoso, Koen Krämer, Markus Stäuble, Dhionis Sako, Fabian Jenelten, Marko Bjelonic, and Marco Hutter. Alma-articulated locomotion and manipula-

tion for a torque-controllable robot. In *ICRA*, 2019. 2.1, 2.4

[9] Nikolai Bernstein. The co-ordination and regulation of movements. *The co-ordination and regulation of movements*, 1966. 2.1

[10] Gerardo Bledt, Matthew J. Powell, Benjamin Katz, Jared Di Carlo, Patrick M Wensing, and Sangbae Kim. Mit cheetah 3: Design and control of a robust, dynamic quadruped robot. In *IROS*, 2018. 2.4

[11] Kenneth Blomqvist, Michel Breyer, Andrei Cramariuc, Julian Förster, Margarita Grinvald, Florian Tschopp, Jen Jen Chung, Lionel Ott, Juan Nieto, and Roland Siegwart. Go fetch: Mobile manipulation in unstructured environments. *ICRA Workshop*, 2020. 2.4

[12] Michaela Bruton and Nicholas O'Dwyer. Synergies in coordination: a comprehensive overview of neural, computational, and behavioral approaches. *Journal of Neurophysiology*, 120(6):2761–2774, 2018. 2.1

[13] Joel Chestnutt, Manfred Lau, German Cheung, James Kuffner, Jessica Hodgins, and Takeo Kanade. Footstep planning for the honda asimo humanoid. In *ICRA*, 2005. 2.4

[14] Ignasi Clavera, Anusha Nagabandi, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*, 2019. 2.4

[15] Michele Colledanchise and Petter Ogren. *Behavior Trees in Robotics and AI: An Introduction*. 07 2018. ISBN 9781138593732. doi: 10.1201/9780429489105. 3.1, 3.2, 3.3.3

[16] Xilun Ding and Fan Yang. Study on hexapod robot manipulation using legs. *Robotica*, 34(2):468–481, February 2016. 2.4, 3.2

[17] Ke Dong, Karime Pereida, Florian Shkurti, and Angela P Schoellig. Catch the ball: Accurate high-speed motions for mobile manipulators via inverse dynamics learning. In *IROS*, 2020. 2.4

[18] Boston Dynamics. Spot Arm. `https://www.bostondynamics.com/`. 2.1

[19] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization based full body control for the atlas robot. In *International Conference on Humanoid Robots*, 2014. 2.4

[20] Kevin French, Shiyu Wu, Tianyang Pan, Zheming Zhou, and Odest Chadwicke Jenkins. Learning behavior trees from demonstration. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7791–7797, May 2019. 3.2

[21] Zipeng Fu, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Minimizing energy consumption leads to the emergence of gaits in legged robots. In *CoRL*,

2021. (document), 2.1, 2.2, 2.4, 3.1, 3.2

[22] Zipeng Fu, Ashish Kumar, Ananye Agarwal, Haozhi Qi, Jitendra Malik, and Deepak Pathak. Coupling vision and proprioception for navigation of legged robots. In *CVPR*, 2022. 2.4

[23] Zipeng Fu, Xuxin Cheng, and Deepak Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023. 3.1, 3.3.2

[24] Hartmut Geyer, Andre Seyfarth, and Reinhard Blickhan. Positive force feedback in bouncing gaits? *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 2003. 2.1, 2.4

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90. 3.3.3

[26] Daniel Honerkamp, Tim Welschehold, and Abhinav Valada. Learning kinematic feasibility for mobile manipulation through deep reinforcement learning. *RA-L*, 2021. 2.4

[27] Marco Hutter, Christian Gehring, Dominic Jud, Andreas Lauber, C Dario Bellicoso, Vassilios Tsounis, Jemin Hwangbo, Karen Bodie, Peter Fankhauser, Michael Bloesch, et al. Anymal-a highly mobile and dynamic quadrupedal robot. In *IROS*, 2016. 2.1, 2.4

[28] Jemin Hwangbo, Joonho Lee, Alexey Dosovitskiy, Dario Bellicoso, Vassilios Tsounis, Vladlen Koltun, and Marco Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 2019. 2.4, A.3

[29] Dong Jin Hyun, Jongwoo Lee, SangIn Park, and Sangbae Kim. Implementation of trot-to-gallop transition and subsequent gallop on the mit cheetah i. *IJRR*, 2016. 2.4

[30] Chieko Sarah Imai, Minghao Zhang, Yuchen Zhang, Marcin Kierebinski, Ruihan Yang, Yuzhe Qin, and Xiaolong Wang. Vision-guided quadrupedal locomotion in the wild with multi-modal delay randomization. *arXiv:2109.14549*, 2021. 2.4

[31] Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, pages 1–1, 2022. 3.1, 3.2, 3.3.2

[32] Yandong Ji, Zhongyu Li, Yinan Sun, Xue Bin Peng, Sergey Levine, Glen Berseth, and Koushil Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. August 2022. 3.2

[33] Aaron M Johnson and Daniel E Koditschek. Legged self-manipulation. *IEEE*

Access, 1:310–334, 2013. 3.1

[34] Aaron M Johnson, Thomas Libby, Evan Chang-Siu, Masayoshi Tomizuka, Robert J Full, and Daniel E Koditschek. Tail assisted dynamic self righting. In *Adaptive Mobile Robotics*. World Scientific, 2012. 2.4

[35] Satoshi Kataoka, Seyed Kamyar Seyed Ghasemipour, Daniel Freeman, and Igor Mordatch. Bi-manual manipulation and attachment via sim-to-real reinforcement learning. *arXiv preprint arXiv:2203.08277*, 2022. 2.2

[36] Mahdi Khoramshahi, Hamed Jalaly Bidgoly, Soroosh Shafiee, Ali Asaei, Auke Jan Ijspeert, and Majid Nili Ahmadabadi. Piecewise linear spine for speed–energy efficiency trade-off in quadruped robots. *Robotics and Autonomous Systems*, 2013. 2.4

[37] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 2016. 2.4

[38] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *Robotics: Science and Systems*, 2021. 3.1, 3.1, 3.1, 3.2, 3.3.1, 3.3.2, **??**, 3.4.1

[39] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots. In *RSS*, 2021. 2.2.2, 3, **??**, 2.4

[40] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. Adapting rapid motor adaptation for bipedal robots. *arXiv preprint arXiv:2205.15299*, 2022. 3.3.2

[41] Mark L Latash. The bliss (not the problem) of motor abundance (not redundancy). *Experimental brain research*, 217(1):1–5, 2012. 2.1

[42] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, October 2020. 3.1

[43] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 2020. 2.2.2, 2.4

[44] Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2018. 3.2

[45] Youngwoon Lee, Joseph J Lim, Anima Anandkumar, and Yuke Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv:2111.07999*, 2021. 3.2

[46] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *JMLR*, 2016. 2.2.2

[47] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2811–2817, 2021. doi: 10.1109/ICRA48506.2021.9560769. 3.1, 3.2

[48] Zhongyu Li, Xuxin Cheng, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for robust parameterized locomotion control of bipedal robots. In *ICRA*, 2021. 2.4

[49] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Robust and versatile bipedal jumping control through multi-task reinforcement learning. *arXiv preprint arXiv:2302.09450*, 2023. 3.1

[50] Kevin Michael Lynch. *Nonprehensile robotic manipulation: Controllability and planning*. Carnegie Mellon University, 1996. 3.1

[51] Yuntao Ma, Farbod Farshidian, Takahiro Miki, Joonho Lee, and Marco Hutter. Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators. *RA-L*, 2022. 2.1, 2.1, 2.4

[52] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU-Based physics simulation for robot learning. August 2021. A.3

[53] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 3.4

[54] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Gait-conditioned policies yield diversified quadrupedal agility. *Conference on Robot Learning*, 2022. 3.1, 3.2

[55] Gabriel B Margolis, Tao Chen, Kartik Paigwar, Xiang Fu, Donghyun Kim, Sang bae Kim, and Pulkit Agrawal. Learning to jump from pixels. In *CoRL*, 2021. 2.2.2

[56] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *RSS*, 2022. 2.2.1

[57] Alejandro Marzinotto, Michele Colledanchise, Christian Smith, and Petter Ögren. Towards a unified behavior trees framework for robot control. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5420–5427,

May 2014. 3.2

[58] Takahiro Miki, Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science Robotics*, January 2022. 2.2.2, 3.1, 3.2

[59] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *IJRR*, 1984. 2.1, 2.4

[60] Ofir Nachum, Michael Ahn, Hugo Ponte, Shixiang Gu, and Vikash Kumar. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019. 3.2

[61] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *ICRA*, 2011. 3.4

[62] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011. 2.3.3, A.6

[63] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 07 2020. doi: 10.15607/RSS.2020.XVI.064. 3.1, 3.2

[64] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Edward Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. In *RSS*, 2020. 2.4

[65] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. AMP: adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, 40(4):1–20, July 2021. 3.1, 3.2

[66] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. ASE: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Trans. Graph.*, 41(4):1–17, July 2022. 3.1, 3.2

[67] Karl Pertsch, Ruta Desai, Vikash Kumar, Franziska Meier, Joseph J. Lim, Dhruv Batra, and Akshara Rai. Cross-domain transfer via semantic skill imitation. *6th Conference on Robot Learning*, 2022. 3.1

[68] AFV Pets. Dogs and cats opening doors, 2018. Available online at: `https://youtu.be/MGGFNO3TVpg`. 3.1

[69] Nicolaus A Radford, Philip Strawser, Kimberly Hambuchen, Joshua S Mehling, William K Verdeyen, A Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, et al. Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics*, 2015. 2.4

[70] Marc H. Raibert. Hopping in legged systems—modeling and simulation for the

two-dimensional one-legged case. *IEEE Transactions on Systems, Man, and Cybernetics*, 1984. 2.1, 2.4

[71] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *CoRL*, 2022. 2.2.1, 3.1, 3.2

[72] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. July 2017. 3.3.1

[73] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347*, 2017. 2.2.1, A.4

[74] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *ICRA*, 2006. 2.1

[75] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2021. 3.3.3

[76] Fan Shi, Timon Homberger, Joonho Lee, Takahiro Miki, Moju Zhao, Farbod Farshidian, Kei Okada, Masayuki Inaba, and Marco Hutter. Circus ANYmal: A quadruped learning dexterous manipulation with its limbs. November 2020. 3.2

[77] Laura Smith, J Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *ICRA*, 2022. 2.4

[78] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. Rapidly adaptable legged robots via evolutionary meta-learning. In *IROS*, 2020. 2.4

[79] Koushil Sreenath, Hae-Won Park, Ioannis Poulakakis, and Jessy W Grizzle. A compliant hybrid zero dynamics controller for stable, efficient and fast bipedal walking on mabel. *IJRR*, 2011. 2.1, 2.4

[80] Charles Sun, Jedrzej Orbik, Coline Manon Devin, Brian H Yang, Abhishek Gupta, Glen Berseth, and Sergey Levine. Fully autonomous real-world reinforcement learning with applications to mobile manipulation. In *CoRL*, 2022. 2.4

[81] Huahua Wang and Arindam Banerjee. Bregman alternating direction method of multipliers. *NeurIPS*, 2014. 2.2.2

[82] Xingxing Wang. Unitree go1. `https://www.unitree.com/products/go1/`. 2.1, 2.3.1, A.6

[83] Luca Weihs, Unnat Jain, Iou-Jen Liu, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alex Schwing. Bridging the imitation gap by adaptive insubordination. *NeurIPS*, 2021. 2.2.2

[84] Josiah Wong, Albert Tung, Andrey Kurenkov, Ajay Mandlekar, Li Fei-Fei, Silvio Savarese, and Roberto Martín-Martín. Error-aware imitation learning from

teleoperation data for mobile manipulation. In *CoRL*, 2022. 2.4

[85] Fei Xia, Chengshu Li, Roberto Martín-Martín, Or Litany, Alexander Toshev, and Silvio Savarese. Relmogen: Leveraging motion generation in reinforcement learning for mobile manipulation. *arXiv preprint arXiv:2008.07792*, 2020. 2.4

[86] Chenyu Yang, Bike Zhang, Jun Zeng, Ayush Agrawal, and Koushil Sreenath. Dynamic legged manipulation of a ball through multi-contact optimization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7513–7520. IEEE, 2020. 3.2

[87] Ruihan Yang, Minghao Zhang, Nicklas Hansen, Huazhe Xu, and Xiaolong Wang. Learning vision-guided quadrupedal locomotion end-to-end with cross-modal transformers. In *ICLR*, 2022. 2.4

[88] Yuxiang Yang, Tingnan Zhang, Erwin Coumans, Jie Tan, and Byron Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783. PMLR, 2022. 3.1, 3.2

[89] KangKang Yin, Kevin Loken, and Michiel Van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics*, 2007. 2.1, 2.4

[90] Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *RSS*, 2017. 2.4

[91] Wenhao Yu, C. Karen Liu, and Greg Turk. Policy transfer with strategy optimization. In *ICLR*, 2018. 2.4

[92] Wenhao Yu, Visak C. V. Kumar, Greg Turk, and C. Karen Liu. Sim-to-real transfer for biped locomotion. In *IROS*, 2019. 2.4

[93] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *RA-L*, 2020. 2.4

[94] Wenxuan Zhou, Lerrel Pinto, and Abhinav Gupta. Environment probing interaction policies. In *ICLR*, 2019. 2.4

[95] Simon Zimmermann, Roi Poranne, and Stelian Coros. Go fetch!-dynamic grasps using boston dynamics spot with external robotic arm. In *ICRA*, 2021. 2.1, 2.4