

Foundation Control Model for General Embodied Intelligence

Wenli Xiao

CMU-RI-TR-25-37

April 6, 2025



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

Thesis Committee:

Dr. Guanya Shi, *chair*

Dr. John Dolan, *chair*

Dr. Changliu Liu

Xiaofeng Guo

*Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Robotics.*

Copyright © 2025 Wenli Xiao. All rights reserved.

To my supportive advisor and hardworking collaborators.

Abstract

With the growing accessibility of humanoid hardware and rapid advances in foundation models, we are entering an era where achieving general embodied intelligence is within reach—enabling humanoid robots to perform a wide range of tasks in human-centric environments. Despite significant progress in language and vision foundation models, controlling humanoids with high degrees of freedom to perform agile, dexterous, and versatile tasks remains a challenge. In this thesis, we explore pathways toward that goal. We first design a task representation framework aimed at scalable and general humanoid whole-body control. Building on this foundation, we propose a systematic real-to-sim-to-real method for achieving agile humanoid control. Finally, we introduce a universal dynamics learning framework for general-purpose robots, and demonstrate its effectiveness on various wheeled platforms—validating the feasibility of learning a world model for humanoids in the future.

Acknowledgments

The experience in the M.S. Robotics program at Carnegie Mellon University has been truly transformative. I am deeply grateful to my advisors, Prof. Guanya Shi and Prof. John Dolan, for their visionary guidance and unwavering support throughout my two-year research journey. I also sincerely thank Prof. Changliu Liu for serving on my thesis committee and for her encouragement and valuable feedback that helped shape the direction of my work.

I've been fortunate to collaborate closely with an exceptional group of labmates, including Tairan He, Chong Zhang, Haoru Xue, Zhengyi Luo, Dvij Kalaria, Tony Tao, Jiawei Gao, Chaoyi Pan, Xiaofeng Guo, Guanqi He, Zeji Yi, Yuanhang Zhang, Zi Wang, Yitang Li, Haoyang Weng, Yogita Choudhary, Nikhil Sobanbabu, Yichao Zhong, Anoushka Alavilli, Giri Anantharaman, Chaitanya Chawla, Maitham F. Al-Sunni, Kai S. Yun, Haotian Lin, Juan Alvarez-Padilla, Simin Liu, and Yiwei Lyu. Their insights, enthusiasm, and camaraderie made this journey both intellectually rewarding and personally fulfilling.

Beyond CMU, I had the privilege of interning with the NVIDIA GEAR Group, where I was mentored by Dr. Jim Fan and Prof. Yuke Zhu. The collaborative environment there significantly accelerated my growth as a researcher and inspired many ideas that continue to guide my current work. I am also thankful to my wonderful collaborators at NVIDIA, including Toru Lin, Zhengjia Xu, Zhengyu Jiang, Yuqi Xie, Jan Kautz, and You Liang Tan, for their guidance and support.

Finally, I want to express my heartfelt thanks to my parents and family for their unwavering support, which has enabled me to pursue my passions freely and wholeheartedly.

As I begin the next chapter of my journey, pursuing a Ph.D. at CMU, I carry with me deep gratitude for all the people and opportunities that have shaped my path.

Contents

1	Introduction	1
2	General Task Representation Design for Humanoid Whole-body Control	3
2.1	Introduction	3
2.2	Method	6
2.2.1	Goal-Conditioned RL for Humanoid Control	6
2.2.2	Command Space Design for Humanoid Control	7
2.2.3	Motion Retargeting	8
2.2.4	Oracle Policy Training from Large-Scale Human Motions	8
2.2.5	Multi-Mode Versatile Controller via Distillation	9
2.3	Experiment	10
2.3.1	Comparison with Specialists	12
2.3.2	Comparison with Other Generalist Training Methods.	12
2.3.3	Real-World Evaluation	13
2.4	Related Work	16
2.5	Conclusion	17
3	Aligning Simulation and Real-world for Agility	19
3.1	Introduction	19
3.2	Related Works	21
3.2.1	Learning-based Methods for Humanoid Control	21
3.2.2	Offline and Online System Identification for Robotics	22
3.2.3	Residual Learning for Robotics	22
3.3	Pre-training: Learning Agile Humanoid Skills	23
3.3.1	Data Generation: Retargeting Human Video Data	23
3.3.2	Phase-based Motion Tracking Policy Training	24
3.4	Post-training: Training Delta Action Model and Fine-tuning Motion Tracking Policy	26
3.4.1	Data Collection	27
3.4.2	Training Delta Action Model	27
3.4.3	Fine-tuning Motion Tracking Policy under New Dynamics	29
3.4.4	Policy Deployment	30
3.5	Performance Evaluation of ASAP	30
3.5.1	Comparison of Dynamics Matching Capability	32

3.5.2	Comparison of Policy Fine-Tuning Performance	34
3.5.3	Real-World Evaluations	35
3.6	Extensive Studies And Analyses	37
3.6.1	Key Factors in Training Delta Action Models	38
3.6.2	Different Usage of Delta Action Model	39
3.6.3	Does ASAP Fine-Tuning Outperform Random Action Noise Fine-Tuning?	40
3.7	Limitations	42
3.8	Conclusion	43
4	Learning Universal Dynamics Model for Agile and Adaptive Mobility	45
4.1	Introduction	45
4.2	Overview	47
4.3	Model and Data	49
4.3.1	Pre-Training with Massive Simulated Data	49
4.3.2	Robust In-Context Adaptive Dynamics Model	50
4.3.3	Fine-Tuning with state-estimation error reduction	50
4.4	System Design	51
4.4.1	MPPI Controller	51
4.4.2	State Estimation	52
4.4.3	Low-Level Controller	53
4.5	Experiments and Results	54
4.6	Experiment	54
4.6.1	Evaluate Model In-context Adaptation Capability	54
4.6.2	Evaluate Model Capability in the Wild	56
4.6.3	Interpret the expressiveness and robustness of AnyCar	56
	Bibliography	59

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

List of Figures

2.1	HOVER enables versatile humanoid control with a unified multi-mode command space. The versatile multi-mode command space supports kinematic position tracking (blue) , local joint angle tracking (yellow) , and root tracking (purple) . Highlighted boxes indicate active commands being tracked, while the masks (dashed boxes on the right) allow selective activation of different command spaces to accommodate various tasks. . .	4
2.2	Overview of HOVER distillation process. The HOVER policy is distilled from the Oracle policy through proprioception and command masking. The task commands for the student are determined via <i>mode-specific</i> and <i>sparsity-based</i> masks, applied to both upper and lower body motions independently. These masks generate diverse task command modes, refining the student’s inputs. The distillation employs DAgger to align the student’s actions with those of the oracle, optimizing through supervised learning on the oracle’s actions.	5
2.3	Comparison between prior work specialists (blue) and our generalist policy (green) under corresponding modes. The metrics used are: upper/lower joint error (rad), global/local body position error (mm), root velocity error (m/s), and root rotation error (rad). These metrics evaluate how accurately each policy tracks reference motions and joint configurations across different control modes. The modes being tracked (activated) by each mode are colored blue	6
2.4	We assess the tracking accuracy of two multi-mode control policies— HOVER (green) and Multi-Mode RL (purple) —across eight distinct humanoid control modes. The comparison is visualized across four key performance metrics in the radar charts.	14
2.5	Real-World Evaluations on different control modes.	15
2.6	HOVER shows robustness under control mode switches during locomotion and real-time teleoperation tests.	17

3.1	Retargeting Human Video Motions to Robot Motions: (a) Human motions are captured from video. (b) Using TRAM [169], 3D human motion is reconstructed in the SMPL parameter format. (c) A reinforcement learning (RL) policy is trained in simulation to track the SMPL motion. (d) The learned SMPL motion is retargeted to the Unitree G1 humanoid robot in simulation. (e) The trained RL policy is deployed on the real robot, executing the final motion in the physical world. This pipeline ensures the retargeted motions remain physically feasible and suitable for real-world deployment.	20
3.2	Overview of ASAP . (a) Motion Tracking Pre-training and Real Trajectory Collection : With the humanoid motions retargeted from human videos, we pre-train multiple motion tracking policies to roll out real-world trajectories. (b) Delta Action Model Training : Based on the real-world rollout data, we train the delta action model by minimizing the discrepancy between simulation state s_t and real-world state s_t^r . (c) Policy Fine-tuning : We freeze the delta action model, incorporate it into the simulator to align the real-world physics and then fine-tune the pre-trained motion tracking policy. (d) Real-World Deployment : Finally, we deploy the fine-tuned policy directly in the real world without delta action model.	23
3.3	Baselines of ASAP . (a) Model-free RL training. (b) System ID from real to sim using real-world data. (c) Learning delta dynamics model using real-world data. (d) Our proposed method, learning delta action model using real-world data.	28
3.4	Replaying IsaacSim State-Action trajectories in IsaacGym. The upper four panels visualize the Unitree G1 humanoid executing a soccer-shooting motion under four distinct open-loop actions. Corresponding metric curves (bottom) quantify tracking performance. Importantly, our delta action model (ASAP) is trained across multiple motions and is not overfitted to this specific example.	30
3.5	Visual comparisons of motion imitation results across different difficulty levels (Easy, Medium, Hard) for various tasks including Jump Forward, Side Jump, Single Foot Balance, Squat, Step Backward, Step Forward, and Walk.	33
3.6	Visualization of G1 motion tracking before and after ASAP fine-tuning in IsaacGym, IsaacSim and Genesis. Top: LeBron James' "Silencer" motion tracking policy fine-tuning for IsaacGym to IsaacSim. Bottom: <i>single foot balance</i> motion tracking policy fine-tuning for IsaacGym to Genesis. . . .	34

3.7	Visualization of LeBron James’ ”Silencer” motion on the G1 robot before (upper figure enclosed in blue) and after (bottom figure enclosed in red) ASAP policy finetuning. The left half shows the policy finetuning for the in-distribution motions while the right half shows the out-of-distribution ones. After ASAP finetuning, the robot behaves more smoothly and reduces jerky lower-body motions.	36
3.8	We deploy the pretrained policy of a forward jump motion tracking task, challenging the 1.35m-tall Unitree G1 robot for a forward leap over 1m. .	37
3.9	Analysis of dataset size, training horizon, and action norm on the performance of π^Δ . (a) Dataset Size: Mean Per Joint Position Error (MPJPE) is evaluated for both in-distribution (green) and out-of-distribution (blue) scenarios. Increasing dataset size leads to enhanced generalization, evidenced by decreasing errors in out-of-distribution evaluations. Closed-loop MPJPE (red bars) also shows improvement with larger datasets. (b) Training Horizon: Open-loop MPJPE (heatmap) improves across evaluation points as training horizons increase, achieving the lowest error at 1.5s. However, closed-loop MPJPE (red bars) shows a sweet spot at a training horizon of 1.0s, beyond which no further improvements are observed. The red dashed line represents the pretrained baseline without π^Δ fine-tuning. (c) Action Norm: The action norm weight significantly influences performance. Both open-loop and closed-loop MPJPE decrease as the weight increases up to 0.1, achieving the lowest error. However, further increases in the action norm weight result in degradation of open-loop performance, highlighting the trade-off between action smoothness and policy flexibility.	38
3.10	MPJPE comparison over timesteps for fine-tuning methods using delta actionmodel. RL Fine-Tuning achieves the lowest error, while Fixed-Point Iteration and Gradient Search perform worse than the baseline (Before DeltaA) showing the highest error.	40
3.11	MPJPE vs. Noise Level for policies fine-tuned with random action noise. Policies with noise levels $\beta \in [0.025, 0.2]$ show improved performance compared to no fine-tuning. Delta action achieves better tracking precision (126 MPJPE) compared to the best action noise (173 MPJPE).	41
3.12	Visualization of IsaacGym-to-IsaacSim π^Δ output magnitude. We compute the average absolute value of each joint over the 4300-episode dataset. Larger red dots indicate higher values. The results suggest that lower-body motors exhibit a larger discrepancy compared to upper-body joints, with the most significant gap observed in the ankle pitch joint of the G1 humanoid.	42

4.1	Performance of AnyCar and baselines in the wild under state estimation errors. Above: A 10 cm tolerance corridor is set as a checkpoint. Below: each row represents the true trajectory of one method, and each column corresponds to a specific setting for the 1/16 scale car: high speed (2 m/s), towing a box, and replacing the front left tire with a plastic wheel. All settings significantly alter the vehicle dynamics.	46
4.2	AnyCar System Pipeline. Phase 1: We collect 100M data in 4 different simulations for pre-training and 0.02M few-shot real-world data for fine-tuning the model. Phase 2: We pre-train the model with the simulation dataset and enhance prediction robustness through masking, adding noise, and attacking the inputs. We also fine-tune using the fine-tuning dataset. Phase 3: We deploy AnyCar in the wild under state estimation error (using SLAM [31, 104, 105] and VIO [153]) to control different vehicles (1/10 scale, 1/16 scale) with different settings (tow object, 3D-printed wheels) on different terrains.	47
4.3	Comparison of different model structures and data scales. The reported testing error is normalized using the mean and standard deviation of the evaluation dataset.	52
4.4	Visualization of AnyCar’s transformer attention in three real-world settings: (a) low speed at 0.5 m/s, (b) high speed at 2 m/s, and (c) towing an object at 2 m/s, all tracking the same reference trajectory. AnyCar’s transformer consistently focuses on the nearest 50 steps across all settings and adaptively attends to different sections of the track. For example, it attends to the first corner in setting (b) and the second corner in setting (c).	52
4.5	Comparison with baselines in the wild.	55
4.6	Comparison of AnyCar robust training methods. (a) Evaluate prediction error in real-world trajectories. (b) Demonstrate predicted trajectory of different methods.	57

List of Tables

2.1	Command space for priors works on whole-body humanoid control. HOVER covers all the command designs designed by prior works with a unified command space and supports multi-mode control by tracking arbitrary subsets of the command elements.	8
2.2	Reward designs for the oracle policy.	9
2.3	Simulation motion imitation evaluation of HOVER and baselines on dataset $\hat{Q}$. Metrics that are tracked by different modes are highlighted in corresponding colors. Results that are statistically significant are highlighted in bold across 5 random seeds.	10
2.4	Comparison between HOVER and specialists. We only report tracking metrics that are tracked by this mode.	13
2.5	Real-world tracking evaluation on 20 standing motions between prior works specialists and our method. Results that are statistically significant are highlighted in bold across 5 tests.	15
3.1	Reward Terms for Pretraining	27
3.2	Domain Randomizations	28
3.3	Reward Terms for Delta Action Learning	29
3.4	SysID Parameters	32
3.5	Open-loop performance comparison across simulators and motion lengths.	32
3.6	Closed-loop motion imitation evaluation across different simulators. All variants are trained with identical rewards.	33
3.7	Real-world closed-loop performance comparing with and without ASAP finetuning on one in-distribution motion and one out-of-distribution motion.	37
4.1	Indoor Results using Ground Truth State-Estimation (Motion Capture)	53
4.2	Indoor Results with State-Estimation Error (VIO)	55

Chapter 1

Introduction

The pursuit of general intelligence in robotics, which enables versatile control across diverse embodiments and tasks, has emerged as a frontier in embodied AI [56, 60, 92, 111]. While recent advances in vision [23, 182, 195] and language [27, 165] have produced foundation models capable of generalizing across modalities and problem domains, robotics remains constrained by embodiment-specific and task-specific controllers. This limits the scalability and adaptability of robotic systems deployed in the real world.

On the other hand, humanoid robots as designed for general purpose tasks have become more accessible and capable in recent years [21]. The progress in hardware and algorithms brings the potential for traversability [131, 192] and dexterity [40, 89, 90] to do a wide range of loco-manipulation tasks in various scenarios [17, 49, 58, 83, 89, 163]. In addition, with the advancement of parallelized simulation and reinforcement learning infrastructure [107, 113], people can train a humanoid to walk in the real world in minutes [190]. However, training in simulation and deployment in the real world (sim-to-real) for single tasks relies on the fidelity of simulation and extensive reward tuning and physics parameter randomization [59, 78]. Sim-to-real is even more challenging in agile and dextrous control for general tasks for high-DoF humanoid robots [40].

This thesis provides insights toward building a **foundation control model** for general humanoid robot—a control framework that 1) generalizes across robot dynamics and tasks, while 2) enabling agile behavior in complex and dynamic environments. Analogous to how large-scale pretraining in vision and language yields generalist models like CLIP [129] and GPT [11], our goal is to build a scalable, task-agnostic policy architecture grounded

in control-centric embodiment. We explore this vision in three progressive stages, each targeting a key capability of generalist robot control:

1. **Task Representation for Generalization and Scalability.** In [Chapter 2](#), we first study the generalizable task representation for humanoid whole-body tracking across diverse scenarios. We propose a scalable interface for humanoid whole-body motion tracking and an efficient learning framework. We can support various existing tele-operation devices by learning from massive human motion datasets. We show that the learned generalist tracking policy outperforms a specialist policy trained with specific settings.
2. **Systematic Real-to-Sim-to-Real Learning.** The Sim-to-real techniques used in prior works [\[49, 52, 192\]](#) have a fundamental limit which prevents humanoids from tracking more agile motions in the real world. In [Chapter 3](#), we aim to systematically minimize the sim-to-real gap by learning from real-world data. Our method learns a residual policy model to compensate the discrepancy between simulation and real-world behavior, and finally be used for fine-tuning an RL policy for better performance. We demonstrate a successful agile jumping motion in the real-world.
3. **Universal Dynamics Models for General and Agile Control.** Finally we aim to jointly tackle agile and general control for cross-embodiment. In [Chapter 4](#), we study the possibility of learning universal dynamics model for agile and general control. Instead of directly working on high-dimensional humanoid robots, we first build prototype foundation on a lower-dimension wheeled robots. The key idea is to train a transformer-based dynamics model that generalizes across diverse vehicle morphologies and driving tasks. With extensive data collection in diverse simulations, robust model training, and few-shot real-world fine-tuning, our model is able to control robots with different physics property for agile maneuver at zero-shot in the real-world. This work lays a foundation for building universal world model for humanoids in the future.

Chapter 2

General Task Representation Design for Humanoid Whole-body Control

2.1 Introduction

Humanoid is a versatile form factor that supports a wide variety of robotic tasks and applications, including bimanual manipulation [17, 83, 89], bipedal locomotion [87, 88, 131, 196], and agile whole-body control [7, 16, 34, 49, 50, 180, 192]. While showing impressive results, each of these efforts uses a different formulation for whole-body control based on the need for their specific task and scenario. Some use root velocity tracking [87, 196] to support locomotion, some choose joint angle tracking [16, 34] to enable expressive movements, and others use kinematic tracking of selected body keypoints [49, 50] to support teleoperation. Although these approaches are similar in terms of the end goal of motion tracking, they require task-specific controller interface and rewards design. This not only makes the development process repetitive and time-consuming, but also limits the versatility of the resultant whole-body controller. For instance, a robot performing bipedal locomotion on uneven terrain using root velocity tracking [87, 196] would struggle to seamlessly switch to a task requiring precise bi-manual manipulation, where joint angle or end-effector tracking [16, 34, 89] might be necessary. These task-specific dependencies limit versatility, as each controller is restricted to a single mode of control. In addition to motion tracking, many pretrained manipulation policies [73, 120] require operating

2. General Task Representation Design for Humanoid Whole-body Control

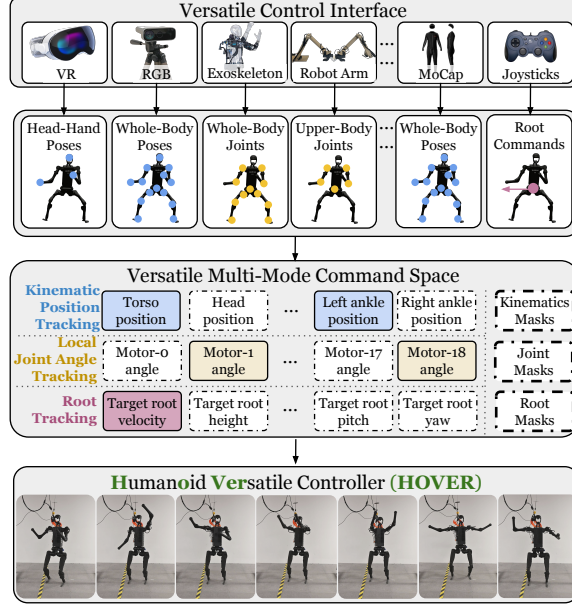


Figure 2.1: HOVER enables versatile humanoid control with a unified multi-mode command space. The versatile multi-mode command space supports **kinematic position tracking (blue)**, **local joint angle tracking (yellow)**, and **root tracking (purple)**. Highlighted boxes indicate active commands being tracked, while the masks (dashed boxes on the right) allow selective activation of different command spaces to accommodate various tasks.

in different configuration spaces, such as joint angles and end-effector positions. This variability highlights the need for a unified low-level humanoid controller capable of adapting to diverse control mode configurations.

Since all these modes are applied to a shared hardware platform, a natural question arises: *Can we create a unified controller that supports all control modes, combining the strengths of each?* This is a non-trivial challenge, as each mode operates within a distinct command space, making direct integration impractical. However, despite differences in control interfaces, the underlying motion objectives often align: stable, human-like motion for humanoid control.

To this end, we present HOVER, a unified neural controller for humanoid whole-body control that supports diverse control modes as shown in Figure 2.1, including over 15 useful modes for real-world applications to a 19-DOF humanoid robot. This versatile command space covers most modes used in prior works [16, 34, 49, 50]. To ensure a robust foundation of motor skills that generalize well across tasks, we train an oracle motion imitator to mimic large-scale human motion data from MoCap [106], covering a wide variety of movements

2. General Task Representation Design for Humanoid Whole-body Control

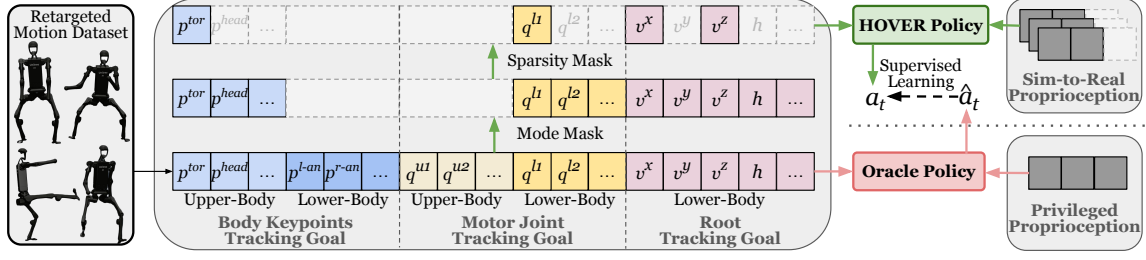


Figure 2.2: **Overview of HOVER distillation process.** The **HOVER policy** is distilled from the **Oracle policy** through proprioception and command masking. The task commands for the student are determined via *mode-specific* and *sparsity-based* masks, applied to both upper and lower body motions independently. These masks generate diverse task command modes, refining the student’s inputs. The distillation employs DAGger to align the student’s actions with those of the oracle, optimizing through supervised learning on the oracle’s actions.

and control objectives. This design choice leverages the inherent adaptability and natural efficiency of human movements, providing the policy with rich motor priors that can be reused across multiple control modes. By grounding the training process in human-like motion, the policy gains a deeper understanding of balance, coordination, and motion control, which are crucial for effective whole-body humanoid behavior. Through a policy distillation process, we transfer these motor skills from the oracle policy into a single “generalist policy” capable of handling multiple control modes. The resulting multi-mode policy not only supports diverse control modes but also *outperforms* policies trained individually for each mode as shown in Figure 2.3. We hypothesize that this is due to the policy leveraging shared physical knowledge across modes, such as maintaining balance, human-like motion, and precise limb control. These shared skills enhance generalization, leading to better performance across all modes. In contrast, single-mode policies often overfit to specific reward structures and training environments, limiting their adaptability. Our multi-mode generalist policy also enables seamless transitions between modes, making it both robust and versatile.

To summarize, our contributions are threefold: 1) we present HOVER, a unified neural controller for humanoid whole-body control supporting multiple control modes; 2) we show that, through policy distillation, HOVER effectively shares motor skills across modes and outperforms individually trained policies; and 3) experiments in both simulation and on a real humanoid robot demonstrate that HOVER achieves seamless transitions between modes and delivers superior multi-mode control compared to other baselines.

2. General Task Representation Design for Humanoid Whole-body Control

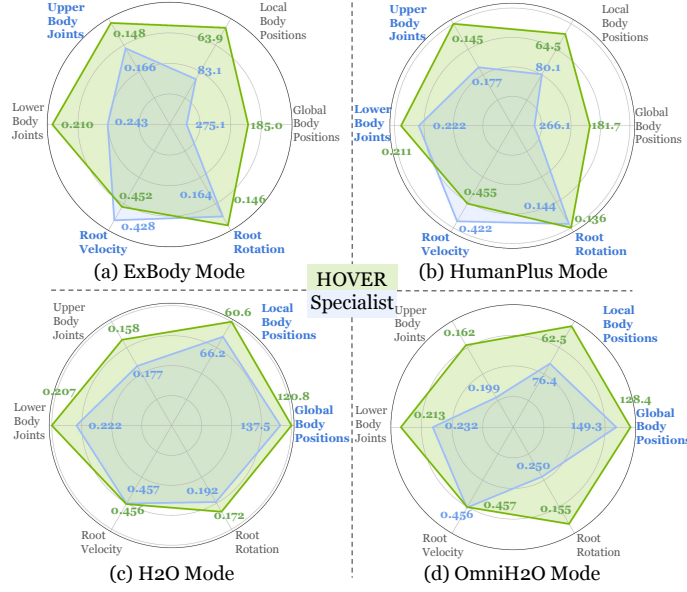


Figure 2.3: Comparison between **prior work specialists (blue)** and our **generalist policy (green)** under corresponding modes. The metrics used are: upper/lower joint error (rad), global/local body position error (mm), root velocity error (m/s), and root rotation error (rad). These metrics evaluate how accurately each policy tracks reference motions and joint configurations across different control modes. The modes being tracked (activated) by each mode are colored **blue**.

2.2 Method

2.2.1 Goal-Conditioned RL for Humanoid Control

We formulate our problem as a goal-conditioned reinforcement learning (RL) task, where the policy π is trained to track real-time human motion. The state s_t comprises both the agent’s proprioception s_t^p and the target goal state s_t^g . The goal state s_t^g provides a unified representation of the target motion goal, which we will describe in detail in [Section 2.2.2](#). Using the agent’s proprioception s_t^p and the goal state s_t^g , we define the reward $r_t = \mathcal{R}(s_t^p, s_t^g)$ for policy optimization. The action $a_t \in \mathbb{R}^{19}$ represents the target joint positions, which are fed into the PD controller to actuate the robot’s degrees of freedom. We employ the proximal policy optimization (PPO) algorithm [142] to maximize the cumulative discounted reward $\mathbb{E} \left[\sum_{t=1}^T \gamma^{t-1} r_t \right]$. This setup is framed as a command-tracking task, where the humanoid learns to follow the target commands at each timestep.

2.2.2 Command Space Design for Humanoid Control

In legged locomotion, root velocity [110] or position tracking [53] is a commonly employed command space. However, focusing solely on root tracking imposes limitations on the full capabilities of humanoid robots, especially for whole-body loco-manipulation tasks. We observe that while prior works [16, 34, 49, 50] have introduced control modes with varying advantages and disadvantages, each is typically tailored to specific subsets of tasks, thus lacking the flexibility required for general-purpose humanoid control. In contrast, our goal is to design a comprehensive control framework that accommodates a wide range of scenarios and is adaptable to various humanoid tasks. To achieve this, the command space must be constructed to satisfy the following key criteria:

- **Generality:** The command space should encompass most existing configurations, allowing a general-purpose controller to replace task-specific controllers without sacrificing performance or versatility. And the space should be sufficiently expressive to interface with real-world control devices, including joysticks, keyboards, motion capture systems, exoskeletons, and virtual reality (VR) headsets as shown in Figure 2.1.
- **Atomicity:** The command space should be composed of independent dimensions, enabling arbitrary combinations of control options to support various modes.

Based on these criteria, we define a unified command space for humanoid whole-body control. This space consists of two primary control regions—upper-body and lower-body control—and incorporates three distinct control modes:

- **Kinematic Position Tracking:** target 3D positions of key rigid body points on the robot.
- **Local Joint Angle Tracking:** target joint angles for each robot motor.
- **Root Tracking:** target root velocity, height, and orientation, specified by roll, pitch, and yaw angles.

In our framework, as shown in Figure 2.1, a one-hot masking vector is introduced to specify which components of the command space are activated for tracking. Recent work on learning-based humanoid whole-body control [16, 34, 49, 50], as shown in Table 2.1 can be viewed as subsets of our unified command space, each representing specific configurations.

2. General Task Representation Design for Humanoid Whole-body Control

Table 2.1: Command space for priors works on whole-body humanoid control. HOVER covers all the command designs designed by prior works with a unified command space and supports multi-mode control by tracking arbitrary subsets of the command elements.

Controller	Multi-Mode Control	Upper-Body Command		Lower-Body Command		
		Kinematic Position	Joint Angle	Kinematic Position	Joint Angle	Root
ExBody [16]	✗	✗	✓	✗	✗	✓
H2O [50]	✗	✓	✗	✓	✗	✗
OmniH2O [49]	✗	✓	✗	✗	✗	✗
HumanPlus [34]	✗	✗	✓	✗	✓	✓
MHC [30]	✓	✗	✓	✗	✓	✓
HOVER (ours)	✓	✓	✓	✓	✓	✓

2.2.3 Motion Retargeting

Recent works have shown the advantage of learning robust whole-body control for humanoid robots from large motion datasets [16, 34, 49, 50]. The retargeting procedure from human motion dataset [106] to humanoid motion dataset has three steps: **Step-1**: We first compute the keypoints positions of the humanoid using forward kinematics, mapping its joint configurations to workspace coordinates. **Step-2**: Next, we fit the SMPL model to match the humanoid’s kinematics by optimizing the SMPL parameters to align with the computed keypoints from forward kinematics. **Step-3**: Finally, the AMASS dataset is retargeted by matching corresponding keypoints between the fitted SMPL model and the humanoid with gradient descent. We follow the same motion retargeting and “sim-to-data” procedure with [50] to convert the large-scale human motion dataset [106] to dataset $\hat{Q}$ that only contains feasible motions for humanoids.

2.2.4 Oracle Policy Training from Large-Scale Human Motions

State Space Design We train an oracle motion imitator $\pi^{\text{oracle}}(a_t | s_t^{\text{p-oracle}}, s_t^{\text{g-oracle}})$. The proprioception is defined as $s_t^{\text{p-oracle}} \triangleq [\mathbf{p}_t, \boldsymbol{\theta}_t, \dot{\mathbf{p}}_t, \boldsymbol{\omega}_t, \mathbf{a}_{t-1}]$, which contains the humanoid rigid-body position $\mathbf{p}_t$, orientation $\boldsymbol{\theta}_t$, linear velocity $\dot{\mathbf{p}}_t$, angular velocity $\boldsymbol{\omega}_t$, and the previous action $\mathbf{a}_{t-1}$. The goal state is defined as $s_t^{\text{g-oracle}} \triangleq [\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\mathbf{p}}_{t+1} - \mathbf{p}_t, \hat{\mathbf{v}}_{t+1} - \mathbf{v}_t, \hat{\boldsymbol{\omega}}_{t+1} - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_t, \hat{\mathbf{p}}_t]$, which contains the reference pose $(\hat{\boldsymbol{\theta}}_t, \hat{\mathbf{p}}_t)$ and one-frame difference between the reference and current state for all rigid bodies of the humanoid. We use the same policy network structure with [49], a three-layer MLP with layer dimensions of [512, 256, 128].

2. General Task Representation Design for Humanoid Whole-body Control

Table 2.2: Reward designs for the oracle policy.

Term	Weight	Term	Weight
Penalty			
Torque limits	-2	DoF position limits	$-1.25e^2$
Termination	$-2.5e^2$	DoF velocity limits	$-5e^1$
Regularization			
DoF acceleration	$-1.1e^{-5}$	DoF velocity	$-4e^{-3}$
Lower Action rate	-3	Upper Action rate	$-6.25e^{-1}$
Torque	$-1e^{-4}$	Feet orientation	$-6.25e^1$
Feet air time	$1e^3$	Feet contact force	$-7.5e^{-1}$
Stumble	$-1.25e^3$	Slippage	$-7.5e^1$
In the air	$-2e^2$	Max feet height per step	$-3e^3$
Task Reward			
DoF position	$3.2e^1$	DoF velocity	$1.6e^1$
Body position	$8e^1$	Body rotation	$2e^1$
Body velocity	8	Body angular velocity	8
Root velocity	$1e^2$	Root rotation	$2e^1$

Reward Design and Domain Randomizations We formulate the reward r_t as the sum of three components: 1) penalty, 2) regularization, and 3) task rewards, detailed in Table 2.2. We follow the same domain randomization in [49] to randomize the physical parameters of the simulated environment and humanoids for successful sim-to-real transfer.

2.2.5 Multi-Mode Versatile Controller via Distillation

Proprioception For the student policy $\pi^{\text{student}}(s_t^{\text{p-student}}, s_t^{\text{g-student}})$ distilled from the oracle teacher π^{oracle} , the proprioception is defined as $s_t^{\text{p-student}} \triangleq [q, \dot{q}, \omega^{\text{base}}, g]_{t-25:t} \cup [a_{t-25:t-1}]$, where q is the joint position, $\dot{q}$ is the joint velocity, ω^{base} is the base angular velocity, g is the gravity vector, and a is the action history. Following [49], we stack these terms over the last 25 steps to represent the student’s proprioceptive input.

Command Mask As illustrated in Figure 2.2, the task command input for the student policy is defined using mode-based and sparsity-based masking. Specifically, the student’s task command input, $s_t^{\text{g-student}}$, is represented as

$$s_t^{\text{g-student}} \triangleq M_{\text{sparsity}} \odot \left[M_{\text{mode}} \odot s_t^{\text{g-upper}}, M_{\text{mode}} \odot s_t^{\text{g-lower}} \right]$$

. The mode mask, M_{mode} , selects a specific task command mode for the upper and lower body independently. For instance, the upper body may track kinematic positions, while the

2. General Task Representation Design for Humanoid Whole-body Control

Table 2.3: Simulation motion imitation evaluation of HOVER and baselines on dataset $\hat{Q}$. Metrics that are tracked by different modes are highlighted in corresponding colors. Results that are statistically significant are highlighted in bold across 5 random seeds.

Method	Kinematic Position					Joint Angle		Root				
	Survive $\uparrow$	$E_{g-mpipe} \downarrow$	$E_{mpipe} \downarrow$	$E_{acc} \downarrow$	$E_{vel} \downarrow$	$E_{upper-j} \downarrow$	$E_{lower-j} \downarrow$	$E_{root-vel} \downarrow$	$E_{root-h} \downarrow$	$E_{root-r} \downarrow$	$E_{root-p} \downarrow$	$E_{root-y} \downarrow$
Oracle policy	99.3% ± 0.203	119 ± 0.442	59.4 ± 0.234	2.63 ± 0.008	5.43 ± 0.024	0.153 ± 0.001	0.206 ± 0.001	0.456 ± 0.002	0.066 ± 0.001	0.065 ± 0.001	0.083 ± 0.001	0.282 ± 0.002
ExBody Mode - Upper: joint angle tracking, Lower: root tracking												
ExBody (Specialist)	99.1% ± 0.212	275 ± 1.650	83.1 ± 0.499	2.63 ± 0.007	6.31 ± 0.034	0.166 ± 0.002	0.243 ± 0.003	0.428 ± 0.007	0.074 ± 0.001	0.070 ± 0.001	0.147 ± 0.002	0.276 ± 0.003
HOVER(Ours)	99.1% ± 0.220	185 ± 1.110	63.9 ± 0.384	3.01 ± 0.009	6.06 ± 0.033	0.148 ± 0.002	0.210 ± 0.004	0.452 ± 0.005	0.063 ± 0.001	0.068 ± 0.001	0.091 ± 0.001	0.279 ± 0.002
HumanPlus Mode - Upper: joint angle tracking, Lower: joint angle tracking, root tracking												
HumanPlus (Specialist)	98.4% ± 0.259	266 ± 1.597	80.1 ± 0.481	2.53 ± 0.007	6.16 ± 0.033	0.177 ± 0.002	0.222 ± 0.002	0.422 ± 0.006	0.061 ± 0.001	0.080 ± 0.001	0.124 ± 0.001	0.228 ± 0.002
HOVER(Ours)	98.9% ± 0.285	182 ± 1.093	64.5 ± 0.387	2.85 ± 0.008	5.91 ± 0.032	0.145 ± 0.001	0.211 ± 0.002	0.455 ± 0.007	0.067 ± 0.001	0.069 ± 0.001	0.104 ± 0.001	0.237 ± 0.002
H2O Mode - Upper: kinematic position tracking (left/right hand, left/right shoulder, left/right elbow), Lower: kinematic position tracking (left/right ankle)												
H2O (Specialist)	99.2% ± 0.233	137 ± 0.827	66.3 ± 0.398	2.63 ± 0.008	5.75 ± 0.028	0.177 ± 0.003	0.221 ± 0.002	0.457 ± 0.004	0.078 ± 0.001	0.067 ± 0.001	0.095 ± 0.001	0.415 ± 0.003
HOVER(Ours)	98.9% ± 0.276	121 ± 0.726	60.6 ± 0.361	2.73 ± 0.008	5.49 ± 0.028	0.158 ± 0.002	0.207 ± 0.002	0.456 ± 0.005	0.065 ± 0.001	0.067 ± 0.001	0.086 ± 0.001	0.365 ± 0.003
OmniH2O Mode - Upper: kinematic position tracking (head, left/right hand), Lower: N/A												
OmniH2O (Specialist)	99.0% ± 0.301	149 ± 0.897	76.4 ± 0.459	2.69 ± 0.007	6.18 ± 0.037	0.199 ± 0.002	0.232 ± 0.003	0.456 ± 0.002	0.071 ± 0.001	0.070 ± 0.001	0.125 ± 0.002	0.306 ± 0.002
HOVER(Ours)	99.0% ± 0.297	128 ± 0.768	62.5 ± 0.368	2.69 ± 0.008	5.65 ± 0.032	0.162 ± 0.002	0.213 ± 0.002	0.457 ± 0.004	0.065 ± 0.001	0.068 ± 0.001	0.089 ± 0.001	0.310 ± 0.002

lower body focuses on joint angle and root tracking, as shown in Figure 2.2. After the mode-specific masking, the sparsity mask, M_{sparsity} , is applied. For example, in some scenarios, the upper body may track only the kinematic positions of the hands, while the lower body tracks only the joint angles of the torso. Every bit of the mode and sparsity binary mask is from a Bernoulli distribution $\mathcal{B}(0.5)$. Mode and sparsity masks are randomized at the episode beginning and remain fixed until the episode ends

Policy Distillation We perform policy distillation using the DAgger framework [138]. For each episode, we roll out the student policy $\pi^{\text{student}}(\mathbf{a}_t | s_t^{\text{p-student}}, s_t^{\text{g-student}})$ in simulation to obtain trajectories of $(s_t^{\text{p-student}}, s_t^{\text{g-student}})$. At each timestep, we also compute the corresponding oracle states $(s_t^{\text{p-oracle}}, s_t^{\text{g-oracle}})$. Using these oracle states, we query the oracle teacher policy $\pi^{\text{oracle}}(\hat{\mathbf{a}}_t | s_t^{\text{p-oracle}}, s_t^{\text{g-oracle}})$ to obtain the reference action $\hat{\mathbf{a}}_t$. The student policy π^{student} is then updated by minimizing the loss function: $\mathcal{L} = \|\hat{\mathbf{a}}_t - \mathbf{a}_t\|_2^2$, where $\hat{\mathbf{a}}_t$ is the reference action from the oracle, and $\mathbf{a}_t$ is the action taken by the student policy.

2.3 Experiment

In this section, we present extensive experimental results in both IsaacGym [107] and the real-world Unitree H1 [160] robot to address the following questions:

- **Q1:** Can HOVER as a generalist policy outperform policies trained for a specific command configuration?
- **Q2:** Can HOVER outperform other methods of training a multi-mode humanoid controller?
- **Q3:** Can HOVER transfer to real-world hardware and execute versatile multi-mode control?

Experiment Setup To answer these questions, we evaluate HOVER on motion tracking in both simulation (Section 2.3.1 and Section 2.3.2) and real-world settings (Section 2.3.3). In simulation, we evaluate using the retargeted AMASS dataset $\hat{Q}$. In the real world, we test 20 standing motion sequences focusing on quantitative tracking and locomotion tasks for qualitative multi-mode control. Our real robot employs a 19-DOF Unitree H1 platform [160] with a total mass of around 51.5kg and a height of around 1.8m.

Baselines To address **Q1** and **Q3**, we compare HOVER with several specialists. As shown in Table 2.1, ExBody [16] focuses on tracking upper body joint angles and root velocity, HumanPlus [34] tracks whole-body joints and root velocity, H2O [50] tracks the kinematic positions of eight keypoints (shoulders, elbows, hands, ankles), and OmniH2O [49] tracks the kinematic positions of the head and both hands. We also compare other useful tracking modes (e.g., left-hand mode, right-hand mode, two-hand mode, head mode). For each control mode, we provide only the relevant observation input to the controller and train the specialist baseline with RL. For instance, in left-hand-only mode, only reference motion of the left hand is provided. To address **Q2**, we compare with another multi-mode RL policy, which follows the same masking process on the goal commands, but trains the baseline with RL objective from scratch. During the multi-mode RL baseline training, mode and sparsity are randomized at the beginning of each episode and remain fixed until the episode ends, which is the same as the randomized masking process during distillation.

Metrics We report survival rate, where the episode terminates if the humanoid hits the ground, not by feet. We calculate tracking error in terms of kinematic pose, joint angles, and root twist and rotations. The mean values of the metrics are computed across all motion sequences from dataset $\hat{Q}$. We evaluate policy’s ability to imitate the reference motion by compare the tracking error of the global body position $E_{g\text{--mpjpe}}$ (mm), the root-relative

mean per-joint (MPJPE) E_{mpjpe} (mm), joint tracking error E_j (rad), root velocity $E_{\text{root-vel}}$ (m/s), and root orientation tracking error $E_{\text{root-ropy}}$ (rad). To show physical realism, we report average joint acceleration E_{acc} (mm/frame²), and velocity E_{vel} (mm/frame) error. To better show the correspondence between control modes and metrics, we highlight the metrics that are actively tracked by each mode with corresponding colors in Table 2.3, Table 2.4 and Table 2.5. For instance, in Table 2.3, upper joint and root metrics are colored with corresponding mode for ExBody mode.

2.3.1 Comparison with Specialists

Comparison with Specialists of Prior Work’s Control Mode To address **Q1** (*Can HOVER as a generalist policy outperform policies trained for a specific command configuration?*), we compare the performance of the same HOVER policy across different control modes against the corresponding specialist policies. For example, the performance of HOVER under ExBody mode is evaluated with a fixed mask to match ExBody mode across the entire dataset $\hat{Q}$. As shown in Table 2.3 and Figure 2.3, HOVER consistently demonstrates superior generalization. In every command mode, HOVER outperforms prior work specialist controllers in at least 7 out of the 12 metrics, as highlighted by the bold values in Table 2.3. This consistent advantage across various control modes underscores the versatility of HOVER. Furthermore, this means that even when focusing on a single control mode without considering multi-mode versatility, distilling from an oracle policy still surpasses RL-trained specialists.

Comparison with Other Specialists of Other Useful Control Mode In addition to the aforementioned baselines, we also evaluate four additional modes: left-hand mode, right-hand mode, two-hand mode, and head mode. We train four RL specialists to track these modes individually. The results in Table 2.4 show that HOVER consistently outperforms specialists in terms of tracking metrics that are trained for specific command configurations.

2.3.2 Comparison with Other Generalist Training Methods.

To address **Q2** (*does HOVER outperform other methods of training a multi-mode humanoid controller?*), we compare HOVER with a multi-mode RL baseline that follows the same masking process on the commands but trains with RL objective from scratch. In Figure 2.4,

Table 2.4: Comparison between HOVER and specialists. We only report tracking metrics that are tracked by this mode.

Method	$E_{g\text{-mpjpe-mode}} \downarrow$	$E_{mpjpe-mode} \downarrow$	$E_{acc-mode} \downarrow$	$E_{vel-mode} \downarrow$
Left Hand Mode - Upper: kinematic position tracking (left hand), Lower: N/A				
Specialist	189 ± 1.526	147 ± 1.324	5.82 ± 0.029	11.2 ± 0.089
HOVER(Ours)	138 ± 1.025	151 ± 0.934	5.45 ± 0.031	10.3 ± 0.104
Right Hand Mode - Upper: kinematic position tracking (right hand), Lower: N/A				
Specialist	220 ± 1.345	216 ± 1.451	6.77 ± 0.051	12.5 ± 0.152
HOVER(Ours)	128 ± 0.774	141 ± 0.821	5.83 ± 0.049	10.8 ± 0.129
2 Hands Mode - Upper: kinematic position tracking (left-right hands), Lower: N/A				
Specialist	137 ± 0.998	145 ± 1.010	5.72 ± 0.037	11.2 ± 0.004
HOVER(Ours)	120 ± 0.901	119 ± 0.827	5.60 ± 0.045	10.1 ± 0.134
Head Mode - Upper: kinematic position tracking (robot head), Lower: N/A				
Specialist	186 ± 1.149	104 ± 0.814	2.22 ± 0.008	6.63 ± 0.065
HOVER(Ours)	133 ± 0.849	80.0 ± 0.711	2.31 ± 0.011	6.40 ± 0.029

we assess tracking error across four metrics: root orientation, upper joint angle, and local and global body positions, measured in eight different modes. We scale the tracking error via $\frac{E^{\max} - E^{(\cdot)}}{E^{\max} - E^{\min}}$ for visualization, where larger radar webs indicate better tracking performance. The results show that HOVER achieves consistently lower tracking error across 32/32 metrics and modes. This performance boost underscores the importance of distilling from an oracle policy that tracks full-body kinematics for learning a generalist whole-body controller.

2.3.3 Real-World Evaluation

To address **Q3** (*does HOVER transfer to real-world hardware and execute versatile multi-mode control?*), we conduct quantitative tracking experiments and locomotion tests for qualitative multi-mode control.

Standing Motion Evaluations We evaluate HOVER’s performance in the real world by tracking 20 different standing motions in $\hat{Q}$. Among these, two motions are visually illustrated in [Figure 2.5](#) (left). The quantitative metrics presented in [Table 2.5](#) demonstrate HOVER outperforms specialist policies in 11 out of 12 metrics. Moreover, we demonstrate successful tracking of root pitch motion, as shown in the middle of [Figure 2.5](#), and full-body

2. General Task Representation Design for Humanoid Whole-body Control



Figure 2.4: We assess the tracking accuracy of two multi-mode control policies—**HOVER (ours)** (green) and **Multi-Mode RL** (purple)—across eight distinct humanoid control modes. The comparison is visualized across four key performance metrics in the radar charts.

2. General Task Representation Design for Humanoid Whole-body Control

Table 2.5: Real-world tracking evaluation on 20 standing motions between prior works specialists and our method. Results that are statistically significant are highlighted in bold across 5 tests.

Method	$E_{g-mpjpe} \downarrow$	$E_{mpjpe} \downarrow$	$E_{upper-j} \downarrow$	$E_{root-ipy} \downarrow$
ExBody Mode				
ExBody (Specialist)	51.3 ± 0.279	39.3 ± 0.214	0.131 ± 0.001	0.036 ± 0.001
HOVER(Ours)	48.9 ± 0.470	36.8 ± 0.201	0.126 ± 0.001	0.032 ± 0.001
HumanPlus Mode				
Specialist	51.0 ± 0.275	36.7 ± 0.202	0.128 ± 0.001	0.035 ± 0.001
HOVER(Ours)	47.4 ± 0.359	35.3 ± 0.194	0.121 ± 0.001	0.038 ± 0.001
OmniH2O Mode				
Specialist	51.2 ± 0.497	42.1 ± 0.233	0.153 ± 0.002	0.040 ± 0.001
HOVER(Ours)	47.5 ± 0.261	41.0 ± 0.227	0.145 ± 0.001	0.037 ± 0.001

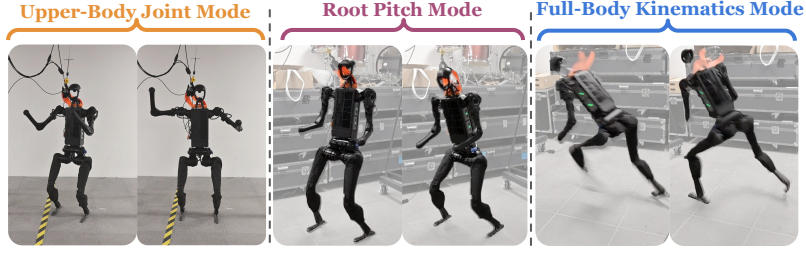


Figure 2.5: Real-World Evaluations on different control modes.

kinematic tracking, as shown on the right of Figure 2.5, where the robot is capable of tracking highly dynamic running motions.

Multi-Mode Evaluations We also evaluate HOVER’s generalization to locomotion in Figure 2.6, where we abruptly switch command modes during operation to simulate real-life scenarios. HOVER successfully transitions from ExBody mode to H2O mode during forward walking in Figure 2.6(a), and from HumanPlus mode to OmniH2O mode while performing turning and backward walking, in Figure 2.6(b). Additionally, we conduct a real-test teleoperation demo with Vision Pro, randomly masking out the positions of the head and hands. For example, in the middle of Figure 2.6(c), the humanoid tracks only the human’s head position, ignoring the waving hands in *head mode*. The results demonstrate that HOVER can smoothly track motions across different modes, showcasing its robustness for real-world scenarios (e.g., when there are occlusions in the reference motions).

2.4 Related Work

Humanoid Whole-Body Controller Performing whole-body control on humanoid hardware is a long-standing challenge in robotics due to the complex structure of humanoid robots. Before the rise in popularity of learning-based controllers, classical humanoid controllers [10, 22, 24, 38, 61, 76, 117, 129, 140, 143] often use a hierarchical model-based optimization to solve for the low-level torque or position commands sent to hardware motors, where actuator-level dynamics on single joints are abstracted to multi-joint [129] or whole-body [140] controllers. Learning-based controllers follow the same design pattern in spirit, where high-level inputs are translated into low-level motor commands via neural networks. The design of controller abstraction and task specification varies by user needs and applications [25, 45, 175]. Recent works on learning-based humanoid whole-body control [16, 34, 49, 50, 144] generally have three design patterns for humanoid whole-body controller: kinematic motion tracking [49, 50, 144], local joint angle tracking [17, 34], and root velocity tracking [87, 196]. Kinematic motion tracking means tracking the full-body kinematic motion for each rigid body of the humanoid, and is heavily inspired by the motion imitation in the graphics community [19, 101, 121, 174]. Local joint angle tracking tracks the local joint angles of the humanoid, which can be considered a specialized case of kinematic motion imitation where global position information is discarded. Root velocity tracking only serves for locomotion ability, and is used for navigation and terrain traversal of humanoids [87, 196]. One can also combine different control modes for upper and lower body: for instance, the upper body can be controlled with local joint angle tracking and the lower body with velocity tracking [16]. Even within the same kinematic tracking pattern, the sparsity design varies depending on the selected keypoints [49, 50]. So far, each of these control modes is independently developed and are not compatible with each other. In this chapter, we aim to unify all of these control modes.

Unified Neural Whole-Body Controller for Humanoid MHC [30, 149] learns multi-mode humanoid controller using RL from retargeted motion, but does not support arbitrary subset of chosen modes and is limited to local joint angles and root tracking. In computer graphics, MaskedMimic [156] enables multi-mode control with flexible kinematic tracking constraint by distillation. Other graphics works leverage reusable motion latent space for downstream flexible control modes [100, 126, 155?]. However, additional policies need to

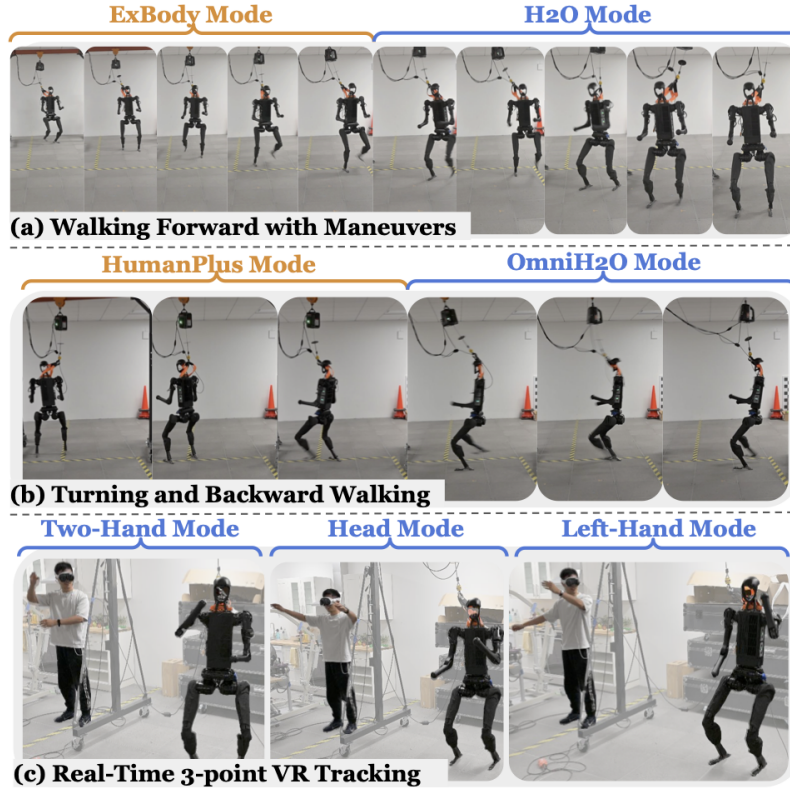


Figure 2.6: HOVER shows robustness under control mode switches during locomotion and real-time teleoperation tests.

be trained. In this chapter, we aim to learn a unified control policy that can be directly used to control real humanoids using different control modes.

2.5 Conclusion

In this chapter, we introduced HOVER, a unified neural controller for humanoid whole-body control that supports diverse control modes. Through the use of a kinematic motion imitator and policy distillation, HOVER consolidates motor skills across multiple control modes into a unified policy that outperforms specialized controllers. Our evaluations collectively illustrate HOVER’s ability to handle diverse real-world control modes, offering superior performance compared to specialist policies. Future work will explore further developing an automated mode-switching module for real-world applications.

2. General Task Representation Design for Humanoid Whole-body Control

Chapter 3

Aligning Simulation and Real-world for Agility

3.1 Introduction

For decades, we have envisioned humanoid robots achieving or even surpassing human-level agility. However, most prior work [39, 86, 87, 94, 130, 131, 173, 196] has primarily focused on locomotion, treating the legs as a means of mobility. Recent studies [16, 49, 50, 52, 65] have introduced whole-body expressiveness in humanoid robots, but these efforts have primarily focused on upper-body motions and have yet to achieve the agility seen in human movement. Achieving agile, whole-body skills in humanoid robots remains a fundamental challenge due to not only hardware limits but also the mismatch between simulated dynamics and real-world physics.

Three main approaches have emerged to bridge the dynamics mismatch: System Identification (SysID) methods, domain randomization (DR), and learned dynamics methods. SysID methods directly estimate critical physical parameters, such as motor response characteristics, the mass of each robot link, and terrain properties [39, 187]. However, these methods require a pre-defined parameter space [93], which may not fully capture the sim-to-real gap, especially when real-world dynamics fall outside the modeled distribution. SysID also often relies on ground truth torque measurements [59], which are unavailable on many widely used hardware platforms, limiting its practical applicability. DR methods,

3. Aligning Simulation and Real-world for Agility

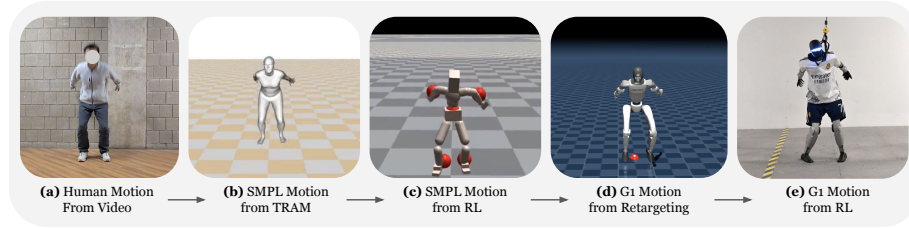


Figure 3.1: Retargeting Human Video Motions to Robot Motions: (a) Human motions are captured from video. (b) Using TRAM [169], 3D human motion is reconstructed in the SMPL parameter format. (c) A reinforcement learning (RL) policy is trained in simulation to track the SMPL motion. (d) The learned SMPL motion is retargeted to the Unitree G1 humanoid robot in simulation. (e) The trained RL policy is deployed on the real robot, executing the final motion in the physical world. This pipeline ensures the retargeted motions remain physically feasible and suitable for real-world deployment.

in contrast, first train control policies in simulation before deploying them on real-world hardware [108, 139, 154]. To mitigate the dynamics mismatch between simulation and real-world physics, DR methods rely on randomizing simulation parameters [123, 158]; but this can lead to overly conservative policies [50], ultimately hindering the development of highly agile skills. Another approach to bridge dynamics mismatch is learning a dynamics model of real-world physics using real-world data. While this approach has demonstrated success in low-dimensional systems such as drones [146] and ground vehicles [178], its effectiveness for humanoid robots remains unexplored.

To this end, we propose **ASAP**, a two-stage framework that aligns the dynamics mismatch between simulation and real-world physics, enabling agile humanoid whole-body skills. **ASAP** involves a pre-training stage where we train base policies in simulation and a post-training stage that finetunes the policy by aligning simulation and real-world dynamics. In the **pre-training** stage, we train a motion tracking policy in simulation using human motion videos as data sources. These motions are first retargeted to humanoid robots [50], and a phase-conditioned motion tracking policy [122] is trained to follow the retargeted movements. However, directly deploying this policy on real hardware results in degraded performance due to the dynamics mismatch. To address this, the **post-training** stage collects real-world rollout data, including proprioceptive states and positions recorded by the motion capture system. The collected data are then replayed in simulation, where the dynamics mismatch manifests as tracking errors. We then train a delta action model that learns to compensate for these discrepancies by minimizing the difference between

real-world and simulated states. This model effectively serves as a residual correction term for the dynamics gap. Finally, we fine-tune the pre-trained policy using the delta action model, allowing it to adapt effectively to real-world physics.

We validate **ASAP** on diverse agile motions and successfully achieve whole-body agility on the Unitree G1 humanoid robot [136]. Our approach significantly reduces motion tracking error compared to prior SysID, DR, and delta dynamics learning baselines in both sim-to-sim (IsaacGym to IsaacSim, IsaacGym to Genesis) and sim-to-real (IsaacGym to Real) transfer scenarios. Our contributions are summarized below.

1. We introduce **ASAP**, a framework that bridges the sim-to-real gap by leveraging a delta action model trained via reinforcement learning (RL) with real-world data.
2. We successfully deploy RL-based whole-body control policies in the real world, achieving previously difficult-to-achieve humanoid motions.
3. Extensive experiments in both simulation and real-world settings demonstrate that **ASAP** effectively reduces dynamics mismatch, enabling highly agile motions on robots and significantly reducing motion tracking errors.
4. To facilitate smooth transfer between simulators, we develop and open-source a multi-simulator training and evaluation codebase for help accelerate further research.

3.2 Related Works

3.2.1 Learning-based Methods for Humanoid Control

In recent years, learning-based methods have made significant progress in whole-body control for humanoid robots. Primarily leveraging reinforcement learning algorithms [142] within physics simulators [107, 113, 159], humanoid robots have learned a wide range of skills, including robust locomotion [39, 84, 85, 87, 88, 130, 131, 179, 193], jumping [86], and parkour [94, 196]. More advanced capabilities, such as dancing [16, 65, 192], locomanipulation [34, 49, 50, 97], and even backflipping [137], have also been demonstrated. Meanwhile, the humanoid character animation community has achieved highly expressive and agile whole-body motions in physics-based simulations [99, 127, 157], including cartwheels [122], backflips [124], sports movements [102, 164, 167, 168, 189], and smooth object interactions [36, 44, 157]. However, transferring these highly dynamic and agile skills to real-world humanoid robots remains challenging due to the dynamics mismatch

between simulation and real-world physics. To address this challenge, our work focuses on learning and compensating for this dynamics mismatch, enabling humanoid robots to perform expressive and agile whole-body skills in the real world.

3.2.2 Offline and Online System Identification for Robotics

The dynamics mismatch between simulators and real-world physics can be attributed to two primary factors: inaccuracies in the robot model descriptions and the presence of complex real-world dynamics that are difficult for physics-based simulators to capture. Traditional approaches address these issues using system identification (SysID) methods [5, 75], which calibrate the robot model or simulator based on real-world performance. These methods can be broadly categorized into *offline SysID* and *online SysID*, depending on whether system identification occurs at test time. *Offline SysID* methods typically collect real-world data and adjust simulation parameters to train policies in more accurate dynamics. The calibration process may focus on modeling actuator dynamics [59, 154, 183], refining robot dynamics models [2, 37, 43, 63, 70], explicitly identifying critical simulation parameters [14, 29, 176, 187], learning a distribution over simulation parameters [4, 54, 133], or optimizing system parameters to maximize policy performance [115, 135]. *Online SysID* methods, in contrast, aim to learn a representation of the robot’s state or environment properties, enabling real-time adaptation to different conditions. These representations can be learned using optimization-based approaches [81, 125, 186, 188], regression-based methods [33, 39, 64, 77, 79, 80, 109, 110, 112, 128, 162, 185], next-state reconstruction techniques [95, 98, 116, 148, 170], direct reward maximization [87], or by leveraging tracking and prediction errors for online adaptation [35, 57, 103, 118]. Our framework takes a different approach from traditional SysID methods by learning a residual action model that directly compensates for dynamics mismatch through corrective actions, rather than explicitly estimating system parameters.

3.2.3 Residual Learning for Robotics

Learning a residual component alongside learned or pre-defined base models has been widely used in robotics. Prior work has explored residual policy models that refine the actions of an initial controller [1, 3, 13, 26, 41, 66, 67, 80, 150]. Other approaches leverage residual components to correct inaccuracies in dynamics models [46, 69, 74, 118, 147] or

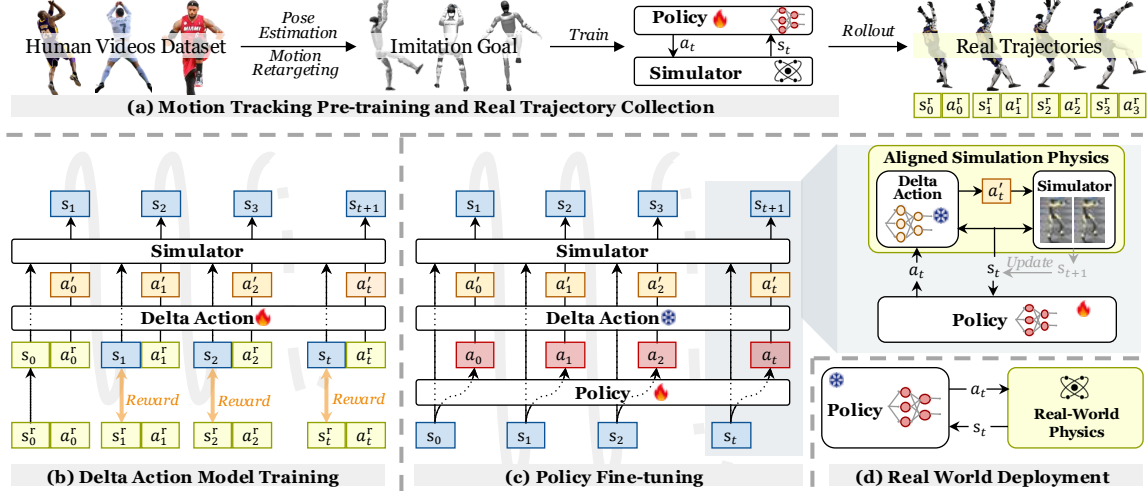


Figure 3.2: Overview of **ASAP**. (a) **Motion Tracking Pre-training and Real Trajectory Collection**: With the humanoid motions retargeted from human videos, we pre-train multiple motion tracking policies to roll out real-world trajectories. (b) **Delta Action Model Training**: Based on the real-world rollout data, we train the delta action model by minimizing the discrepancy between simulation state s_t and real-world state s_t^r . (c) **Policy Fine-tuning**: We freeze the delta action model, incorporate it into the simulator to align the real-world physics and then fine-tune the pre-trained motion tracking policy. (d) **Real-World Deployment**: Finally, we deploy the fine-tuned policy directly in the real world without delta action model.

to model residual trajectories resulting from residual actions [20] for achieving precise and agile motions. RGAT [69] uses a residual action model with a learned forward dynamics to refine the simulator. Our framework builds on this idea by using RL-based residual actions to align the dynamics mismatch between simulation and real-world physics, enabling agile whole-body humanoid skills.

3.3 Pre-training: Learning Agile Humanoid Skills

3.3.1 Data Generation: Retargeting Human Video Data

To track expressive and agile motions, we collect a video dataset of human movements and retarget it to robot motions, creating imitation goals for motion-tracking policies, as shown in Figure 3.1 and Figure 3.2 (a).

Transforming Human Video to SMPL Motions We begin by recording videos (see Figure 3.1 (a) and Figure 3.11) of humans performing expressive and agile motions. Using TRAM [169], we reconstruct 3D motions from videos. TRAM estimates the global trajectory of the human motions in SMPL parameter format [96], which includes global root translation, orientation, body poses, and shape parameters, as shown in Figure 3.1 (b). The resulting motions are denoted as $\mathcal{D}_{\text{SMPL}}$.

Simulation-based Data Cleaning Since the reconstruction process can introduce noise and errors [50], some estimated motions may not be physically feasible, making them unsuitable for motion tracking in the real world. To address this, we employ a “sim-to-data” cleaning procedure. Specifically, we use MaskedMimic [157], a physics-based motion tracker, to imitate the SMPL motions from TRAM in IsaacGym simulator [107]. The motions (Figure 3.1 (c)) that pass this simulation-based validation are saved as the cleaned dataset $\mathcal{D}_{\text{SMPL}}^{\text{Cleaned}}$.

Retargeting SMPL Motions to Robot Motions With the cleaned dataset $\mathcal{D}_{\text{SMPL}}^{\text{Cleaned}}$ in SMPL format, we retarget the motions into robot motions following the shape-and-motion two-stage retargeting process [50]. Since the SMPL parameters estimated by TRAM represent various human body shapes, we first optimize the shape parameter β' to approximate a humanoid shape. By selecting 12 body links with correspondences between humans and humanoids, we perform gradient descent on β' to minimize joint distances in the rest pose. Using the optimized shape β' along with the original translation p and pose θ , we apply gradient descent to further minimize the distances of the body links. This process ensures accurate motion retargeting and produces the cleaned robot trajectory dataset $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$, as shown in Figure 3.1 (d).

3.3.2 Phase-based Motion Tracking Policy Training

We formulate the motion-tracking problem as a goal-conditioned reinforcement learning (RL) task, where the policy π is trained to track the retargeted robot movement trajectories in the dataset $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$. Inspired by [122], the state s_t includes the robot’s proprioception s_t^p and a time phase variable $\phi \in [0, 1]$, where $\phi = 0$ represents the start of a motion and $\phi = 1$ represents the end. This time phase variable alone ϕ is proven to be sufficient to serve

as the goal state s_t^g for single-motion tracking [122]. The proprioception s_t^p is defined as $s_t^p \triangleq [\mathbf{q}_{t-4:t}, \dot{\mathbf{q}}_{t-4:t}, \boldsymbol{\omega}_{t-4:t}^{root}, \mathbf{g}_{t-4:t}, \mathbf{a}_{t-5:t-1}]$, with 5-step history of joint position $\mathbf{q}_t \in \mathbb{R}^{23}$, joint velocity $\dot{\mathbf{q}}_t \in \mathbb{R}^{23}$, root angular velocity $\boldsymbol{\omega}_t^{root} \in \mathbb{R}^3$, root projected gravity $\mathbf{g}_t \in \mathbb{R}^3$, and last action $\mathbf{a}_{t-1} \in \mathbb{R}^{23}$. Using the agent’s proprioception s_t^p and the goal state s_t^g , we define the reward as $r_t = \mathcal{R}(s_t^p, s_t^g)$, which is used for policy optimization. The specific reward terms can be found in Table 3.1. The action $\mathbf{a}_t \in \mathbb{R}^{23}$ corresponds to the target joint positions and is passed to a PD controller that actuates the robot’s degrees of freedom. To optimize the policy, we use the proximal policy optimization (PPO) [142], aiming to maximize the cumulative discounted reward $\mathbb{E} \left[\sum_{t=1}^T \gamma^{t-1} r_t \right]$. We identify several design choices that are crucial for achieving stable policy training:

Asymmetric Actor-Critic Training Real-world humanoid control is inherently a partially observable Markov decision process (POMDP), where certain task-relevant properties that are readily available in simulation become unobservable in real-world scenarios. However, these missing properties can significantly facilitate policy training in simulation. To bridge this gap, we employ an asymmetric actor-critic framework, where the critic network has access to privileged information such as the global positions of the reference motion and the root linear velocity, while the actor network relies solely on proprioceptive inputs and a time-phase variable. This design not only enhances phase-based motion tracking during training but also enables a simple, phase-driven motion goal for sim-to-real transfer. Crucially, because the actor does not depend on position-based motion targets, our approach eliminates the need for odometry during real-world deployment—overcoming a well-documented challenge in prior work on humanoid robots [49, 50].

Termination Curriculum of Tracking Tolerance Training a policy to track agile motions in simulation is challenging, as certain motions can be too difficult for the policy to learn effectively. For instance, when imitating a jumping motion, the policy often fails early in training and learns to remain on the ground to avoid landing penalties. To mitigate this issue, we introduce a termination curriculum that progressively refines the motion error tolerance throughout training, guiding the policy toward improved tracking performance. Initially, we set a generous termination threshold of 1.5m, meaning the episode terminates if the robot deviates from the reference motion by this margin. As training progresses, we gradually tighten this threshold to 0.3m, incrementally increasing the tracking demand on

the policy. This curriculum allows the policy to first develop basic balancing skills before progressively enforcing stricter motion tracking, ultimately enabling successful execution of high-dynamic behaviors.

Reference State Initialization Task initialization plays a crucial role in RL training. We find that naively initializing episodes at the start of the reference motion leads to policy failure. For example, in Cristiano Ronaldo’s jumping training, starting the episode from the beginning forces the policy to learn sequentially. However, a successful backflip requires mastering the landing first—if the policy cannot land correctly, it will struggle to complete the full motion from takeoff. To address this, we adopt the Reference State Initialization (RSI) framework [122]. Specifically, we randomly sample time-phase variables between 0 and 1, which effectively randomizes the starting point of the reference motion for the policy to track. We then initialize the robot’s state based on the corresponding reference motion at that phase, including root position and orientation, root linear and angular velocities and joint positions and velocities. This initialization strategy significantly improves motion tracking training, particularly for agile whole-body motions, by allowing the policy to learn different motion phases in parallel rather than being constrained to a strictly sequential learning process.

Reward Terms We define the reward function r_t with the sum of three terms: 1) penalty, 2) regularization, and 3) task rewards. A detailed summary of these components is provided in Table 3.1.

Domain Randomizations To improve the robustness of the pre-trained policy in Figure 3.2 (a), we utilized basic domain randomization techniques listed in Table 3.2.

3.4 Post-training: Training Delta Action Model and Fine-tuning Motion Tracking Policy

The policy trained in the first stage can track the reference motion in the real-world but does not achieve high motion quality. Thus, during the second stage, as shown in Figure 3.2 (b) and (c), we leverage real-world data rolled out by the pre-trained policy to train a delta

Table 3.1: Reward Terms for Pretraining

Term	Weight	Term	Weight
Penalty			
DoF position limits	-10.0	DoF velocity limits	-5.0
Torque limits	-5.0	Termination	-200.0
Regularization			
Torques	-1×10^{-6}	Action rate	-0.5
Feet orientation	-2.0	Feet heading	-0.1
Slippage	-1.0		
Task Reward			
Body position	1.0	VR 3-point	1.6
Body position (feet)	2.1	Body rotation	0.5
Body angular velocity	0.5	Body velocity	0.5
DoF position	0.75	DoF velocity	0.5

action model, followed by policy refinement through dynamics compensation using this learned delta action model.

3.4.1 Data Collection

We deploy the pretrained policy in the real world to perform whole-body motion tracking tasks (as depicted in Figure 3.8) and record the resulting trajectories, denoted as $\mathcal{D}^r = \{s_0^r, a_0^r, \dots, s_T^r, a_T^r\}$, as illustrated in Figure 3.2 (a). At each timestep t , we use a motion capture device and onboard sensors to record the state: $s_t = [p_t^{\text{base}}, v_t^{\text{base}}, \alpha_t^{\text{base}}, \omega_t^{\text{base}}, q_t, \dot{q}_t]$, where $p_t^{\text{base}} \in \mathbb{R}^3$ represents the robot base 3D position, $v_t^{\text{base}} \in \mathbb{R}^3$ is base linear velocity, $\alpha_t^{\text{base}} \in \mathbb{R}^4$ is the robot base orientation represented as a quaternion, $\omega_t^{\text{base}} \in \mathbb{R}^3$ is the base angular velocity, $q_t \in \mathbb{R}^{23}$ is the vector of joint positions, and $\dot{q}_t \in \mathbb{R}^{23}$ represents joint velocities.

3.4.2 Training Delta Action Model

Due to the sim-to-real gap, when we replay the real-world trajectories in simulation, the resulting simulated trajectory will likely deviate significantly from real-world recorded

Table 3.2: Domain Randomizations

Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.2, 1.1)$
P Gain	$\mathcal{U}(0.925, 1.05) \times \text{default}$
Control delay	$\mathcal{U}(20, 40)\text{ms}$
External Perturbation	
Push robot	interval = 10s, $v_{xy} = 0.5 \text{ m/s}$

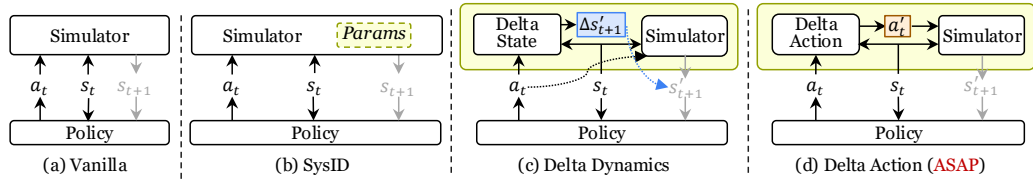


Figure 3.3: Baselines of **ASAP**. (a) Model-free RL training. (b) System ID from real to sim using real-world data. (c) Learning delta dynamics model using real-world data. (d) Our proposed method, learning delta action model using real-world data.

trajectories. This discrepancy is a valuable learning signal for learning the mismatch between simulation and real-world physics. We leverage an RL-based delta/residual action model to compensate for the sim-to-real physics gap.

As illustrated in Figure 3.2 (b), the delta action model is defined as $\Delta a_t = \pi_\theta^\Delta(s_t, a_t)$, where the policy π_θ^Δ learns to output corrective actions based on the current state s_t and the action a_t . These corrective actions (Δa_t) are added to the real-world recorded actions (a_t^r) to account for discrepancies between simulation and real-world dynamics.

The RL environment incorporates this delta action model by modifying the simulator dynamics as follows: $s_{t+1} = f^{\text{sim}}(s_t, a_t^r + \Delta a_t)$ where f^{sim} represents the simulator’s dynamics, a_t^r is the reference action recorded from real-world rollouts, and Δa_t introduces corrections learned by the delta action model.

During each RL step:

1. The robot is initialized at the real-world state s_t^r .
2. A reward signal is computed to minimize the discrepancy between the simulated state s_{t+1} and the recorded real-world state s_{t+1}^r , with an additional action magnitude regularization term $\exp(-\|a_t\|) - 1$, as specified in Table 3.3. The workflow is

Table 3.3: Reward Terms for Delta Action Learning

Term	Weight	Term	Weight
Penalty			
DoF position limits	−10.0	DoF velocity limits	−5.0
Torque limits	−0.1	Termination	−200.0
Regularization			
Action rate	−0.01	Action norm	−0.2
Task Reward			
Body position	1.0	VR 3-point	1.0
Body position (feet)	1.0	Body rotation	0.5
Body angular velocity	0.5	Body velocity	0.5
DoF position	0.5	DoF velocity	0.5

illustrated in [Figure 3.2 \(b\)](#).

3. PPO is used to train the delta action policy π_θ^Δ , learning corrected Δa_t to match simulation and the real world.

By learning the delta action model, the simulator can accurately reproduce real-world failures. For example, consider a scenario where the simulated robot can jump because its motor strength is overestimated, but the real-world robot cannot jump due to weaker motors. The delta action model π_θ^Δ will learn to reduce the intensity of lower-body actions, simulating the motor limitations of the real-world robot. This allows the simulator to replicate the real-world dynamics and enables the policy to be fine-tuned to handle these limitations effectively.

3.4.3 Fine-tuning Motion Tracking Policy under New Dynamics

With the learned delta action model $\pi^\Delta(s_t, a_t)$, we can reconstruct the simulation environment with

$$s_{t+1} = f^{\text{ASAP}}(s_t, a_t) = f^{\text{sim}}(s_t, a_t + \pi^\Delta(s_t, a_t)),$$

As shown in [Figure 3.2 \(c\)](#), we keep the π^Δ model parameters frozen, and fine-tune the pretrained policy with the same reward summarized in [Table 3.1](#).

3. Aligning Simulation and Real-world for Agility

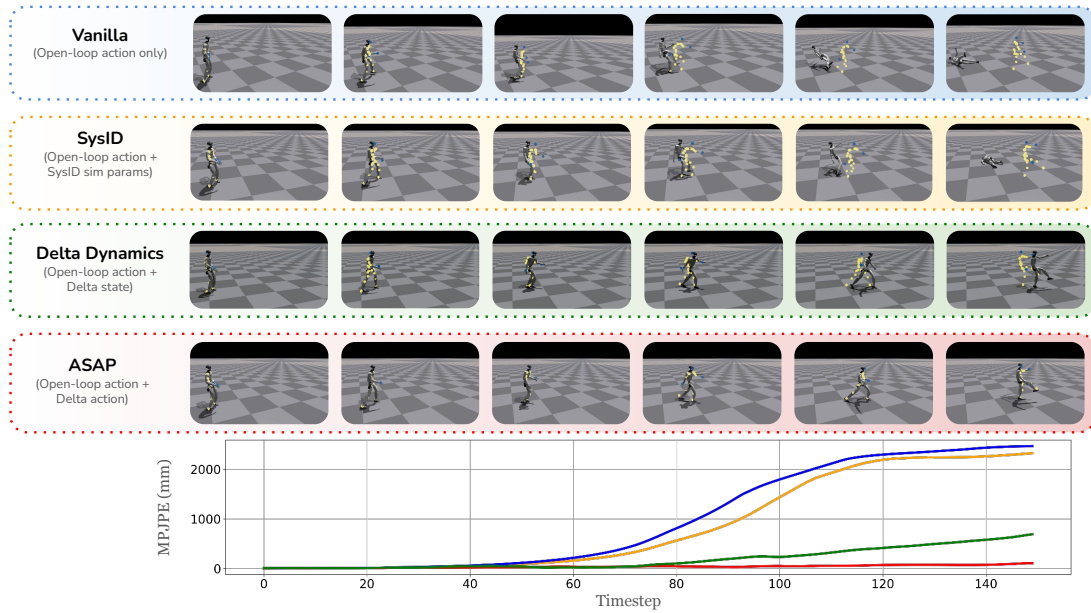


Figure 3.4: Replaying IsaacSim State-Action trajectories in IsaacGym. The upper four panels visualize the Unitree G1 humanoid executing a soccer-shooting motion under four distinct open-loop actions. Corresponding metric curves (bottom) quantify tracking performance. Importantly, our delta action model (ASAP) is trained across multiple motions and is not overfitted to this specific example.

3.4.4 Policy Deployment

Finally, we deploy the fine-tuned policy without delta action model in the real world as shown in Figure 3.2 (d). The fine-tuned policy shows enhanced real-world motion tracking performance compared to the pre-trained policy. Quantitative improvements will be discussed in Section 3.5.

3.5 Performance Evaluation of ASAP

In this section, we present extensive experimental results on three policy transfers: IsaacGym [107] to IsaacSim [113], IsaacGym to Genesis [6], and IsaacGym to real-world Unitree G1 humanoid robot. Our experiments aim to address the following key questions:

- **Q1:** Can ASAP outperform other baseline methods to compensate for the dynamics mismatch?

- **Q2:** Can **ASAP** finetune policy to outperform SysID and Delta Dynamics methods?
- **Q3:** Does **ASAP** work for sim-to-real transfer?

Experiments Setup. To address these questions, we evaluate **ASAP** on motion tracking tasks in both simulation (Section 3.5.1 and Section 3.5.2) and real-world settings (Section 3.5.3).

In the simulation, we use the retargeted motion dataset from the videos we shoot, denoted as $\mathcal{D}_{\text{Robot}}^{\text{Cleaned}}$, which contains diverse human motion sequences. We select 43 motions categorized into three difficulty levels: easy, medium, and hard (as partially visualized in Figure 3.5), based on motion complexity and the required agility. **ASAP** is evaluated through simulation-to-simulation transfer by training policies in IsaacGym and using two other simulators, IsaacSim and Genesis, as a proxy of “real-world” environments. This setup allows for a systematic evaluation of **ASAP**’s generalization and transferability. The success of the transfer is assessed by metrics described in subsequent sections.

For real-world evaluation, we deploy **ASAP** on Unitree G1 robot with fixed wrists to track motion sequences that has obvious sim-to-real gap. These sequences are chosen to capture a broad range of motor capabilities and demonstrate the sim-to-real capability for agile whole-body control.

Baselines. We have the following baselines:

Oracle: This baseline is trained and evaluated entirely within IsaacGym. It assumes perfect alignment between the training and testing environments, serving as an upper bound for performance in simulation.

Vanilla (Figure 3.3 a): The RL policy is trained in IsaacGym and evaluated in IsaacSim, Genesis, or the real world.

SysID (Figure 3.3 b): We identify the following representative parameters in our simulated model that best align the ones in the real world: base center of mass (CoM) shift (c_x, c_y, c_z) , base link mass offset ratio k_m and low-level PD gain ratios (k_p^i, k_d^i) where $i = 1, 2, \dots, 23$. Specifically, we search the best parameters among certain discrete ranges by replaying the recorded trajectories in real with different simulation parameters summarized in Table 3.4. We then finetune the pre-trained policy in IsaacGym with the best SysID parameters.

DeltaDynamics (Figure 3.3 c): We train a residual dynamics model $f_{\theta}^{\Delta}(s_t, a_t)$ to capture the discrepancy between simulated and real-world physics.

Table 3.4: SysID Parameters

Parameter	Range	Parameter	Range
c_x	$[-0.02, 0.0, 0.02]$	c_y	$[-0.02, 0.0, 0.02]$
c_z	$[-0.02, 0.0, 0.02]$	k_m	$[0.95, 1.0, 1.05]$
k_p^i	$[0.95, 1.0, 1.05]$	k_d^i	$[0.95, 1.0, 1.05]$

Table 3.5: Open-loop performance comparison across simulators and motion lengths.

Simulator & Length		IsaacSim				Genesis			
Length	Method	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}
0.25s	OpenLoop	19.5	15.1	6.44	5.80	19.8	15.3	6.53	5.88
	SysID	19.4	15.0	6.43	5.74	19.3	15.0	6.42	5.73
	DeltaDynamics	24.4	13.6	9.43	7.85	20.0	12.4	8.42	6.89
	ASAP	19.9	15.6	6.48	5.86	19.0	14.9	6.19	5.59
0.5s	OpenLoop	33.3	23.2	6.80	6.84	33.1	23.0	6.78	6.82
	SysID	32.1	22.2	6.57	6.56	32.2	22.3	6.57	6.57
	DeltaDynamics	36.5	16.4	8.89	7.98	27.8	14.0	7.63	6.74
	ASAP	26.8	19.2	5.09	5.36	25.9	18.4	4.93	5.19
1.0s	OpenLoop	80.8	43.5	10.6	11.1	82.5	44.5	10.8	11.4
	SysID	77.6	41.5	10.2	10.7	76.5	41.6	10.0	10.5
	DeltaDynamics	68.1	21.5	9.61	9.14	50.2	17.2	8.19	7.62
	ASAP	37.9	22.9	4.38	5.26	36.9	22.6	4.23	5.10

Metrics. We report success rate, deeming imitation unsuccessful when, at any point during imitation, the average difference in body distance is on average further than 0.5m. We evaluate policy’s ability to imitate the reference motion by comparing the tracking error of the global body position $E_{g\text{-mpjpe}}$ (mm), the root-relative mean per-joint (MPJPE) E_{mpjpe} (mm), acceleration error E_{acc} (mm/frame²), and root velocity E_{vel} (mm/frame). The mean values of the metrics are computed across all motion sequences used.

3.5.1 Comparison of Dynamics Matching Capability

To address **Q1** (Can **ASAP** outperform other baseline methods to compensate for the dynamics mismatch?), we establish sim-to-sim transfer benchmarks to assess the effectiveness of different methods in bridging the dynamics gap. IsaacGym serves as the *training environment*, while IsaacSim and Genesis function as *testing environments*. The primary objective is to evaluate the generalization capability of each approach when exposed to new dynamics conditions. *Open-loop* evaluation measures how accurately a method can reproduce testing-environment trajectories in the training environment. This is achieved by

Table 3.6: Closed-loop motion imitation evaluation across different simulators. All variants are trained with identical rewards.

Level	Test Environment	IsaacSim					Genesis				
		Succ $\uparrow$	$E_{g\text{-mpjpe}} \downarrow$	$E_{\text{mpjpe}} \downarrow$	$E_{\text{acc}} \downarrow$	$E_{\text{vel}} \downarrow$	Succ $\uparrow$	$E_{g\text{-mpjpe}} \downarrow$	$E_{\text{mpjpe}} \downarrow$	$E_{\text{acc}} \downarrow$	$E_{\text{vel}} \downarrow$
Easy	Oracle (IsaacGym $\rightarrow$ IsaacGym)	100% $\pm$ 0.000%	97.5 $\pm$ 0.605	43.2 $\pm$ 0.112	2.56 $\pm$ 0.024	4.48 $\pm$ 0.023	100% $\pm$ 0.000%	97.5 $\pm$ 0.605	43.2 $\pm$ 0.112	2.56 $\pm$ 0.024	4.48 $\pm$ 0.023
	Vanilla (IsaacGym $\rightarrow$ TestEnv)	100% $\pm$ 0.000%	107 $\pm$ 0.578	45.4 $\pm$ 0.169	2.83 $\pm$ 0.012	4.59 $\pm$ 0.021	100% $\pm$ 0.000%	140 $\pm$ 1.85	70.1 $\pm$ 0.626	2.68 $\pm$ 0.042	4.65 $\pm$ 0.046
	SysID	100% $\pm$ 0.000%	105 $\pm$ 1.135	47.8 $\pm$ 0.970	3.09 $\pm$ 0.011	4.98 $\pm$ 0.020	100% $\pm$ 0.000%	127 $\pm$ 0.233	79.9 $\pm$ 0.330	2.99 $\pm$ 0.035	4.95 $\pm$ 0.012
	DeltaDynamics	100% $\pm$ 0.000%	127 $\pm$ 2.97	56.7 $\pm$ 0.390	3.50 $\pm$ 0.028	5.56 $\pm$ 0.031	83.3% $\pm$ 0.000%	168 $\pm$ 7.62	87.0 $\pm$ 1.51	3.08 $\pm$ 0.18	5.39 $\pm$ 0.34
	ASAP	100% $\pm$ 0.000%	106 $\pm$ 0.498	44.3 $\pm$ 0.103	2.74 $\pm$ 0.025	4.46 $\pm$ 0.020	100% $\pm$ 0.000%	125 $\pm$ 4.75	73.5 $\pm$ 0.570	2.10 $\pm$ 0.083	4.11 $\pm$ 0.133
Medium	Oracle (IsaacGym $\rightarrow$ IsaacGym)	100% $\pm$ 0.000%	111 $\pm$ 0.635	48.8 $\pm$ 0.133	2.63 $\pm$ 0.017	4.82 $\pm$ 0.019	100% $\pm$ 0.000%	111 $\pm$ 0.635	48.8 $\pm$ 0.133	2.63 $\pm$ 0.017	4.82 $\pm$ 0.019
	Vanilla (IsaacGym $\rightarrow$ TestEnv)	100% $\pm$ 0.000%	114 $\pm$ 0.720	49.2 $\pm$ 0.104	2.92 $\pm$ 0.021	5.07 $\pm$ 0.016	94.3% $\pm$ 7.00%	169 $\pm$ 5.76	72.0 $\pm$ 0.692	3.26 $\pm$ 0.076	5.86 $\pm$ 0.101
	SysID	100% $\pm$ 0.000%	115 $\pm$ 1.256	49.1 $\pm$ 0.560	3.43 $\pm$ 0.021	5.01 $\pm$ 0.017	100% $\pm$ 0.000%	138 $\pm$ 2.70	75.4 $\pm$ 1.18	3.14 $\pm$ 0.042	5.50 $\pm$ 0.058
	DeltaDynamics	83.3% $\pm$ 0.000%	151 $\pm$ 2.62	68.0 $\pm$ 0.364	2.90 $\pm$ 0.047	5.90 $\pm$ 0.107	83.3% $\pm$ 0.000%	190 $\pm$ 1.46	89.4 $\pm$ 0.50	3.44 $\pm$ 0.16	7.49 $\pm$ 0.11
	ASAP	100% $\pm$ 0.000%	112 $\pm$ 1.648	49.3 $\pm$ 0.574	2.53 $\pm$ 0.019	4.45 $\pm$ 0.026	100% $\pm$ 0.000%	126 $\pm$ 1.63	71.2 $\pm$ 0.163	2.81 $\pm$ 0.037	5.13 $\pm$ 0.066
Hard	Oracle (IsaacGym $\rightarrow$ IsaacGym)	100% $\pm$ 0.000%	116 $\pm$ 0.711	52.5 $\pm$ 0.298	3.40 $\pm$ 0.027	6.16 $\pm$ 0.028	100% $\pm$ 0.000%	116 $\pm$ 0.711	52.5 $\pm$ 0.298	3.40 $\pm$ 0.027	6.16 $\pm$ 0.028
	Vanilla (IsaacGym $\rightarrow$ TestEnv)	100% $\pm$ 0.000%	148 $\pm$ 0.845	51.6 $\pm$ 0.137	4.41 $\pm$ 0.055	6.88 $\pm$ 0.064	82.9% $\pm$ 5.70%	175 $\pm$ 9.77	80.7 $\pm$ 1.69	3.87 $\pm$ 0.175	7.19 $\pm$ 0.199
	SysID	100% $\pm$ 0.000%	165 $\pm$ 3.83	58.4 $\pm$ 0.229	4.87 $\pm$ 0.197	7.13 $\pm$ 0.131	100% $\pm$ 0.000%	186 $\pm$ 3.84	93.0 $\pm$ 1.49	4.98 $\pm$ 0.245	8.98 $\pm$ 0.119
	DeltaDynamics	66.7% $\pm$ 0.000%	137 $\pm$ 2.59	60.2 $\pm$ 0.477	4.20 $\pm$ 0.041	7.10 $\pm$ 0.024	60.0% $\pm$ 5.70%	190 $\pm$ 14.0	89.6 $\pm$ 9.34	4.29 $\pm$ 1.16	8.70 $\pm$ 2.23
	ASAP	100% $\pm$ 0.000%	129 $\pm$ 1.57	56.5 $\pm$ 1.15	3.72 $\pm$ 0.036	6.52 $\pm$ 0.042	100% $\pm$ 0.000%	129 $\pm$ 2.31	77.0 $\pm$ 1.07	2.69 $\pm$ 0.040	5.65 $\pm$ 0.073

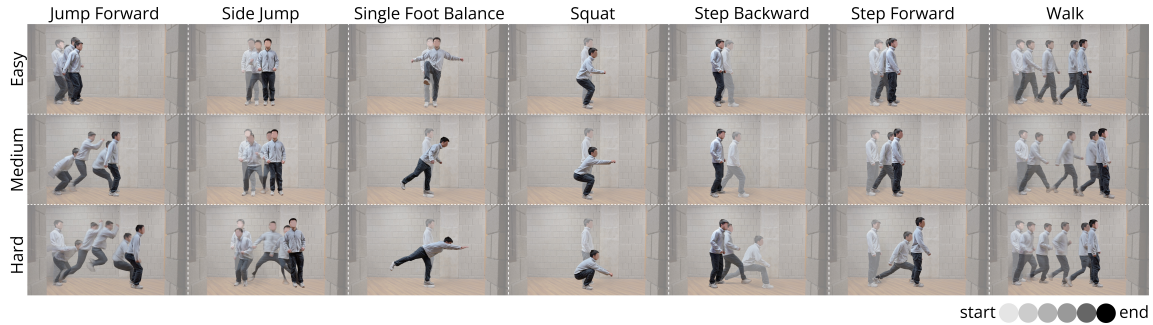


Figure 3.5: Visual comparisons of motion imitation results across different difficulty levels (Easy, Medium, Hard) for various tasks including Jump Forward, Side Jump, Single Foot Balance, Squat, Step Backward, Step Forward, and Walk.

rolling out the same trajectory executed in the testing environment and assessing tracking discrepancies using key metrics such as MPJPE. An ideal method should minimize the discrepancies between training and testing trajectories when replaying testing-environment actions, thereby demonstrating an improved capacity for compensating dynamics mismatch. Quantitative results in Table 3.5 demonstrate that ASAP consistently outperforms the OpenLoop baseline across all replayed motion lengths, achieving lower $E_{g\text{-mpjpe}}$ and E_{mpjpe} values, which indicate improved alignment with testing-env trajectories. While SysID helps address short-horizon dynamics gaps, it struggles with long-horizon scenarios due to cumulative error buildup. DeltaDynamics improves upon both SysID and OpenLoop for long horizons but suffers from overfitting, as evidenced by cascading errors magnified over time, as shown in Figure 3.4. ASAP, however, demonstrates superior generalization by learning residual policies that effectively bridge the dynamics gap. Comparable trends

3. Aligning Simulation and Real-world for Agility

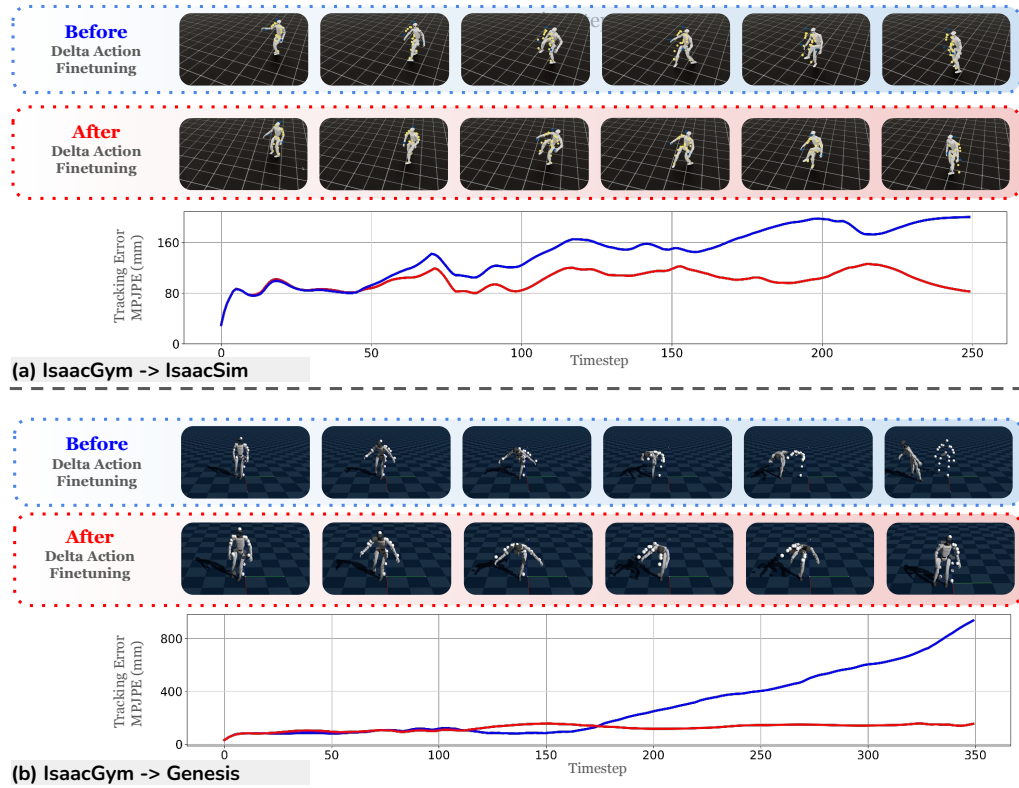


Figure 3.6: Visualization of G1 motion tracking before and after **ASAP** fine-tuning in IsaacGym, IsaacSim and Genesis. Top: LeBron James’ ”Silencer” motion tracking policy fine-tuning for IsaacGym to IsaacSim. Bottom: *single foot balance* motion tracking policy fine-tuning for IsaacGym to Genesis.

are observed in the Genesis simulator, where **ASAP** achieves notable improvements across all metrics relative to the baseline. These results emphasize the efficacy of learning delta action model to reduce the physics gap and improve open-loop replay performance.

3.5.2 Comparison of Policy Fine-Tuning Performance

To address **Q2** (*Can **ASAP** finetune policy to outperform SysID and Delta Dynamics methods?*), we evaluate the effectiveness of different methods in fine-tuning RL policies for improved testing-environment performance. We fine-tune RL policies in modified training environments and subsequently deploy them in the testing environments, quantifying motion-tracking errors in testing environments. As shown in Table 3.6, **ASAP** consistently outperforms baselines such as Vanilla, SysID, and DeltaDynamics across all difficulty

levels (Easy, Medium, and Hard) in both simulators (IsaacSim and Genesis). For the Easy level, our method achieves the lowest $E_{g\text{-mpjpe}}$ and E_{mpjpe} in IsaacSim ($E_{g\text{-mpjpe}} = 106$ and $E_{\text{mpjpe}} = 44.3$) and Genesis ($E_{g\text{-mpjpe}} = 125$ and $E_{\text{mpjpe}} = 73.5$), with minimal acceleration (E_{acc}) and velocity (E_{vel}) errors. In more challenging tasks, such as the Hard level, our method continues to excel, significantly reducing motion-tracking errors. For instance, in Genesis, it achieves $E_{g\text{-mpjpe}} = 129$ and $E_{\text{mpjpe}} = 77.0$, outperforming SysID and DeltaDynamics by substantial margins. Additionally, our method consistently maintains a 100% success rate across both simulators, unlike DeltaDynamics, which experiences lower success rates in harder environments. To further illustrate the advantages of **ASAP**, we provide per-step visualizations in Figure 3.6, comparing **ASAP** with RL policies deployed without fine-tuning. These visualizations demonstrate that **ASAP** successfully adapts to new dynamics and maintains stable tracking performance, whereas baseline methods accumulate errors over time, leading to degraded tracking capability. These results highlight the robustness and adaptability of our approach in addressing the sim-to-real gap while preventing overfitting and exploitation. The findings validate that **ASAP** is an effective paradigm for improving closed-loop performance and ensuring reliable deployment in complex real-world scenarios.

3.5.3 Real-World Evaluations

To answer **Q3** (*Does **ASAP** work for sim-to-real transfer?*). We validate **ASAP** on real-world Unitree G1 robot.

Real-World Data. In the real-world experiments, we prioritize both motion safety and representativeness by selecting five motion-tracking tasks, including (i) *kick*, (ii) *jump forward*, (iii) *step forward and back*, (iv) *single foot balance* and (v) *single foot jump*. However, collecting over 400 real-world motion clips—the minimum required to train the full 23-DoF delta action model in simulation, as discussed in Section 3.4.2—poses significant challenges. Our experiments involve highly dynamic motions that cause rapid overheating of joint motors, leading to hardware failures (two Unitree G1 robots broke during data collection). Given these constraints, we adopt a more sample-efficient approach by focusing exclusively on learning a 4-DoF ankle delta action model rather than the full-body 23-DoF model. This decision is motivated by two key factors: (1) the limited availability of real-world data makes training the full 23-DoF delta action model infeasible,

3. Aligning Simulation and Real-world for Agility

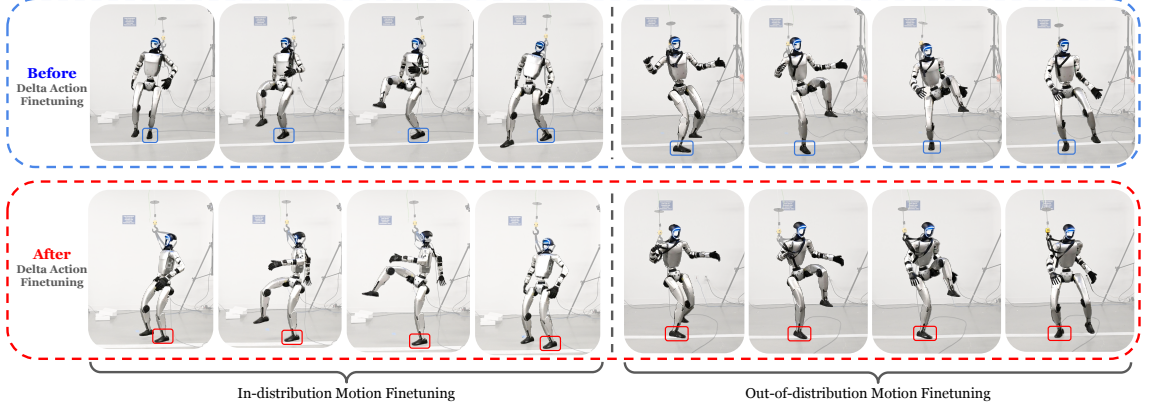


Figure 3.7: Visualization of LeBron James’ ”Silencer” motion on the G1 robot before (upper figure enclosed in blue) and after (bottom figure enclosed in red) ASAP policy finetuning. The left half shows the policy finetuning for the in-distribution motions while the right half shows the out-of-distribution ones. After ASAP finetuning, the robot behaves more smoothly and reduces jerky lower-body motions.

and (2) the Unitree G1 robot [136] features a mechanical linkage design in the ankle, which introduces a significant sim-to-real gap that is difficult to bridge with conventional modeling techniques [71]. Under this setting, the original 23 DoF delta action model reduces to 4 DoF delta action model, which needs much less data to be trainable. In practice, we collect 100 motion clips, which prove sufficient to train an effective 4-DoF delta action model for real-world scenarios.

We execute the tracking policy 30 times for each task. In addition to these motion-tracking tasks, we also collect 10 minutes of locomotion data. The locomotion policy will be addressed in the next section, which is also utilized to bridge different tracking policies.

Policy Transition. In the real world, we cannot easily reset the robot as in simulators, and therefore we train a robust locomotion policy for the policy transition between different motion-tracking tasks. Our locomotion command contains (v, ω, Π) , where v and ω indicate the linear and angular velocities while Π indicates the command to walk or stand still. After each motion-tracking task is done, our locomotion policy will take over to keep the robot balance until the next motion-tracking task begins. In this way, the robot is able to execute multiple tasks without manually resetting.

Real-World Results. The sim-to-real gap is more pronounced than simulator-to-simulator discrepancies due to factors such as noisy sensor input, inaccuracies in robot modeling, and actuator differences. To evaluate the effectiveness of ASAP in addressing

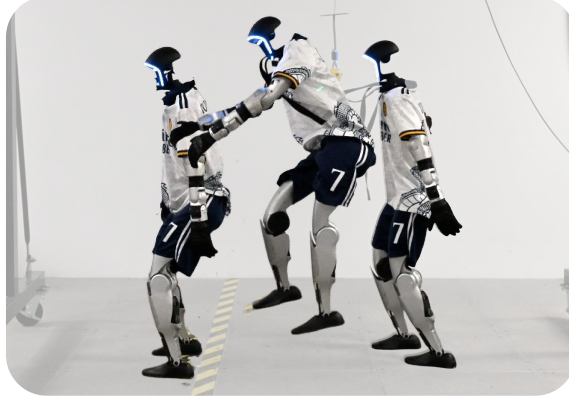


Figure 3.8: We deploy the pretrained policy of a forward jump motion tracking task, challenging the 1.35m-tall Unitree G1 robot for a forward leap over 1m.

Table 3.7: Real-world closed-loop performance comparing with and without **ASAP** finetuning on one in-distribution motion and one out-of-distribution motion.

Motion	Real-World-Kick				Real-World-LeBron (OOD)			
Method	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}	$E_{g\text{-mpjpe}}$	E_{mpjpe}	E_{acc}	E_{vel}
Vanilla	61.2	43.5	2.96	2.91	159	55.3	3.43	6.43
ASAP	50.2	40.1	2.46	2.70	112	47.5	2.84	5.94

these gaps, we compare the closed-loop performance of **ASAP** with the *Vanilla* baseline on two representative motion tracking tasks (kicking and "Silencer") in which observe obvious sim-to-real gaps. To show the generalizability of the learned delta action model for out-of-distribution motions, we also fine-tune the policy for LeBron James' "Silencer" motion as shown in Figure 3.7. The experiment data is summarized in Table 3.7. It shows that **ASAP** outperforms the baseline on both in-distribution and out-of-distribution humanoid motion tracking tasks, achieving a considerable reduction of the tracking errors across all key metrics ($E_{g\text{-mpjpe}}$, E_{mpjpe} , E_{acc} and E_{vel}). These findings highlight the effectiveness of **ASAP** in improving sim-to-real transfer for agile humanoid motion tracking.

3.6 Extensive Studies And Analyses

In this section, we aim to thoroughly analyze **ASAP** by addressing three central research questions:

- **Q4:** How to best train the delta action model of **ASAP**?

3. Aligning Simulation and Real-world for Agility

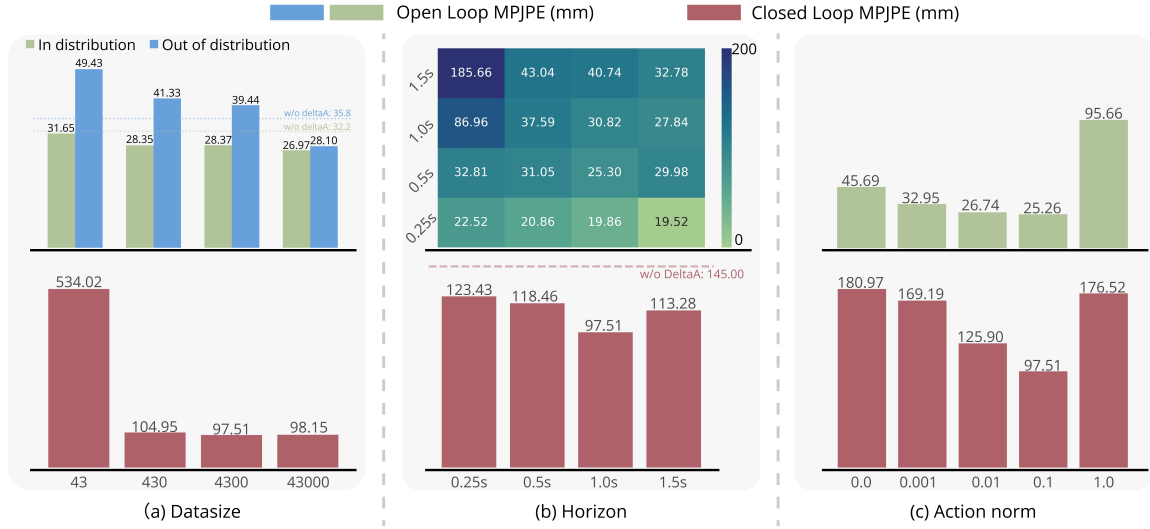


Figure 3.9: Analysis of dataset size, training horizon, and action norm on the performance of π^Δ . (a) **Dataset Size:** Mean Per Joint Position Error (MPJPE) is evaluated for both in-distribution (green) and out-of-distribution (blue) scenarios. Increasing dataset size leads to enhanced generalization, evidenced by decreasing errors in out-of-distribution evaluations. Closed-loop MPJPE (red bars) also shows improvement with larger datasets. (b) **Training Horizon:** Open-loop MPJPE (heatmap) improves across evaluation points as training horizons increase, achieving the lowest error at 1.5s. However, closed-loop MPJPE (red bars) shows a sweet spot at a training horizon of 1.0s, beyond which no further improvements are observed. The red dashed line represents the pretrained baseline without π^Δ fine-tuning. (c) **Action Norm:** The action norm weight significantly influences performance. Both open-loop and closed-loop MPJPE decrease as the weight increases up to 0.1, achieving the lowest error. However, further increases in the action norm weight result in degradation of open-loop performance, highlighting the trade-off between action smoothness and policy flexibility.

- **Q5:** How to best use the delta action model of **ASAP**?
- **Q6:** Why and how does **ASAP** work?

3.6.1 Key Factors in Training Delta Action Models

To Answer **Q4** (How to best train the delta action model of **ASAP**), we conduct a systematic study on key factors influencing the performance of the delta action model. Specifically, we investigate the impact of dataset size, training horizon, and action norm weight, evaluating their effects on both open-loop and closed-loop performance. Our analysis uncovers the essential principles for effectively training a high-performing delta action model.

Dataset Size We analyze the impact of dataset size on the training and generalization of π^Δ . Simulation data is collected in Isaac Sim, and π^Δ is trained in Isaac Gym. Open-loop performance is assessed on both in-distribution (training) and out-of-distribution (unseen) trajectories, while closed-loop performance is evaluated using the fine-tuned policy in Isaac Sim. As shown in Figure 3.9 (a), increasing the dataset size improves π^Δ 's generalization, evidenced by reduced errors in out-of-distribution evaluations. However, the improvement in closed-loop performance saturates, with a marginal decrease of only 0.65% when scaling from 4300 to 43000 samples, suggesting limited additional benefit from larger datasets.

Training Horizon The rollout horizon plays a crucial role in learning π^Δ . As shown in Figure 3.9 (b), longer training horizons generally improve open-loop performance, with a horizon of 1.5s achieving the lowest errors across evaluation points at 0.25s, 0.5s, and 1.0s. However, this trend does not consistently extend to closed-loop performance. The best closed-loop results are observed at a training horizon of 1.0s, indicating that excessively long horizons do not provide additional benefits for fine-tuned policy.

Action Norm Weight Training π^Δ incorporates an action norm reward to balance dynamics alignment and minimal correction. As illustrated in Figure 3.9 (c), both open-loop and closed-loop errors decrease as the action norm weight increases, reaching the lowest error at a weight of 0.1. However, further increasing the action norm weight causes open-loop errors to rise, likely due to the minimal action norm reward dominates in the delta action RL training. This highlights the importance of carefully tuning the action norm weight to achieve optimal performance.

3.6.2 Different Usage of Delta Action Model

To answer **Q5** (How to best use the delta action model of **ASAP?**), we compare multiple strategies: fixed-point iteration, gradient-based optimization, and reinforcement learning (RL). Given a learned delta policy π^Δ such that:

$$f^{\text{sim}}(s, a + \pi^\Delta(s, a)) \approx f^{\text{real}}(s, a),$$

and a nominal policy $\hat{\pi}(s)$ that performs well in simulation, the goal is to fine-tune $\hat{\pi}(s)$ for real-world deployment.

A simple approach is one-step dynamics matching, which leads to the relationship:

$$\pi(s) = \hat{\pi}(s) - \pi^\Delta(s, \pi(s)).$$

We consider two RL-free methods: fixed-point iteration and gradient-based optimization. Fixed-point iteration refines $\hat{\pi}(s)$ iteratively, while gradient-based optimization minimizes a loss function to achieve a better estimate. These methods are compared against RL fine-tuning, which adapts $\hat{\pi}(s)$ using reinforcement learning in simulation.

Our experiments in Figure 3.10 show that RL fine-tuning achieves the lowest tracking error during deployment, outperforming training-free methods. Both RL-free approaches are myopic and suffer from out-of-distribution issues, limiting their real-world applicability.

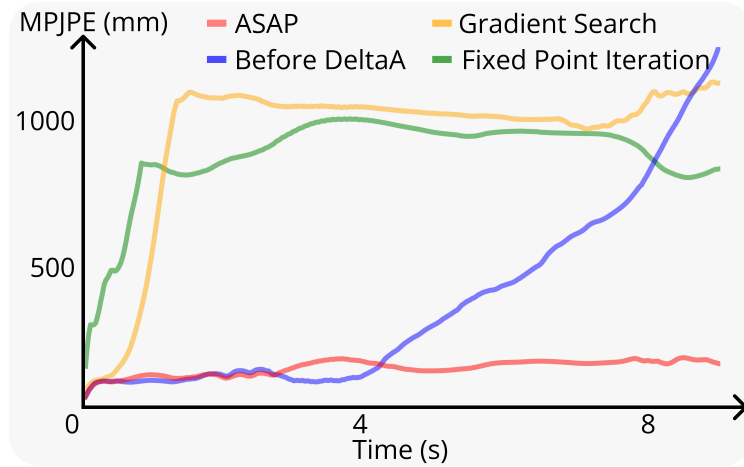


Figure 3.10: MPJPE comparison over timesteps for fine-tuning methods using delta actionmodel. RL Fine-Tuning achieves the lowest error, while Fixed-Point Iteration and Gradient Search perform worse than the baseline (Before DeltaA) showing the highest error.

3.6.3 Does **ASAP** Fine-Tuning Outperform Random Action Noise Fine-Tuning?

To answer Q6 (*How does **ASAP** work?*), we validate **ASAP** finetuning is better than injecting random-action-noise-based finetuning. And we visualize the average magnitude of the delta action model for each joint.

Random torque noise [12] is a widely used domain randomization technique for legged robots. To determine whether delta action facilitates fine-tuning of pre-trained policies

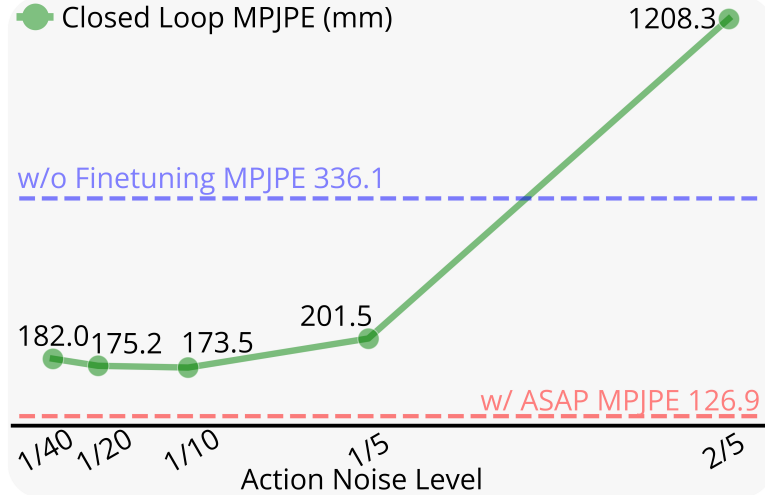


Figure 3.11: MPJPE vs. Noise Level for policies fine-tuned with random action noise. Policies with noise levels $\beta \in [0.025, 0.2]$ show improved performance compared to no fine-tuning. Delta action achieves better tracking precision (126 MPJPE) compared to the best action noise (173 MPJPE).

toward real-world dynamics rather than merely enhancing robustness through random action noise, we analyze its impact. Specifically, we assess the effect of applying random action noise during policy fine-tuning in Isaac Gym by modifying the environment dynamics as $s_{t+1} = f^{\text{sim}}(s_t, a_t + \beta \delta_a)$, where $\delta_a \sim \mathcal{U}[0, 1]$, and deploy it in Genesis. We conduct an ablation study to examine the influence of the noise magnitude, β , varying from 0.025 to 0.4. As shown in Figure 3.11, within the constrained range of $\beta \in [0.025, 0.2]$, policies fine-tuned with action noise outperform those without fine-tuning in terms of global tracking error (MPJPE). However, the performance of the action noise approach (MPJPE of 150) does not match the precision achieved by ASAP (MPJPE of 126). Furthermore, we visualize the average output of π^Δ learned from IsaacSim data in Figure 3.12, which reveals non-uniform discrepancies across joints. For example, in the G1 humanoid robot under our experimental setup, lower-body motors exhibit a larger dynamics gap compared to upper-body joints. Within the lower-body, the ankle and knee joints show the most pronounced discrepancies. Additionally, asymmetries between the left and right body motors further highlight the complexity. Such structured discrepancies cannot be effectively captured by merely adding uniform action noise. These findings, along with the results in Figure 3.4, demonstrate that delta action not only enhances policy robustness but also enables effective adaptation to real-world dynamics, outperforming naive randomization strategies.

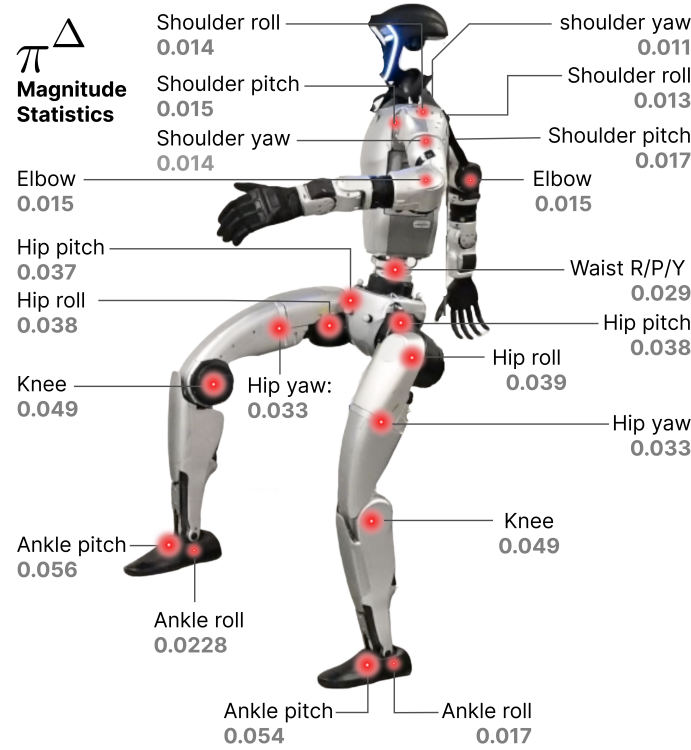


Figure 3.12: Visualization of IsaacGym-to-IsaacSim π^{Δ} output magnitude. We compute the average absolute value of each joint over the 4300-episode dataset. Larger red dots indicate higher values. The results suggest that lower-body motors exhibit a larger discrepancy compared to upper-body joints, with the most significant gap observed in the ankle pitch joint of the G1 humanoid.

3.7 Limitations

While **ASAP** demonstrates promising results in bridging the sim-to-real gap for agile humanoid control, our framework has several real-world limitations that highlights critical challenges in scaling agile humanoid control to real-world:

- **Hardware Constraints:** Agile whole-body motions exert significant stress on robots, leading to motor overheating and hardware failure during data collection. Two Unitree G1 robots were broken to some extent during our experiments. This bottleneck limits the scale and diversity of real-world motion sequences that can be safely collected.
- **Dependence on Motion Capture Systems:** Our pipeline requires MoCap setup to record real-world trajectories. This introduces practical deployment barriers in unstructured environments where MoCap setups are unavailable.

- **Data-Hungry Delta Action Training:** While reducing the delta action model to 4 DoF ankle joints improved sample efficiency, training the full 23 DoF model remains impractical for real-world deployment due to the large demand of required motion clips (e.g., > 400 episodes in simulation for the 23 DoF delta action training).

Future directions could focus on developing damage-aware policy to mitigate hardware risks, leveraging MoCap-free alignment to eliminate the reliance on MoCap, and exploring adaptation techniques for delta action models to achieve sample-efficient few-shot alignment.

3.8 Conclusion

We present **ASAP**, a two-stage framework that bridges the sim-to-real gap for agile humanoid control. By learning a universal delta action model to capture dynamics mismatch, **ASAP** enables policies trained in simulation to adapt seamlessly to real-world physics. Extensive experiments demonstrate significant reductions in motion tracking errors (up to 52.7% in sim-to-real tasks) and successful deployment of diverse agile skills—including agile jumps and kicks—on the Unitree G1 humanoid. Our work advances the frontier of sim-to-real transfer for agile whole-body control, paving the way for versatile humanoid robots in real-world applications.

3. Aligning Simulation and Real-world for Agility

Chapter 4

Learning Universal Dynamics Model for Agile and Adaptive Mobility

4.1 Introduction

The use of the transformer [161] architecture in contemporary robot learning is ubiquitous across perception [28, 91], planning [82] and control [166] tasks. Reinforcement learning with transformers, such as [62] and [15], is used in many downstream applications in bi-manual manipulation [194], navigation [145], humanoid locomotion [132], and whole-body tele-operation [18, 32]. Vision-language-action (VLA) models such as OpenVLA [72], RT-1 [9], and RT-2 [197] demonstrate the scalability of employing transformers in robotics. These models trained on internet-scale data can generalize knowledge and skills to different complex tasks.

Recent advances in robot learning have also introduced more “specialist” systems that can perform highly agile locomotion tasks [47, 48, 51, 191]. In particular, on wheeled robots, previous works have achieved high-speed autonomous driving on racetracks [181], grass fields [152], loose sand [171, 172], and off-road terrain [42, 151]. However, most of these efforts [42, 151] are optimized for specific car models and environments, requiring extensive system identification and model training [152, 171], which is costly to fine-tune and difficult to transfer to other wheeled platforms.

On the other hand, agile wheeled control for safety-critical applications requires precise



Figure 4.1: Performance of AnyCar and baselines in the wild under state estimation errors. Above: A 10 cm tolerance corridor is set as a checkpoint. Below: each row represents the true trajectory of one method, and each column corresponds to a specific setting for the 1/16 scale car: high speed (2 m/s), towing a box, and replacing the front left tire with a plastic wheel. All settings significantly alter the vehicle dynamics.

dynamics modeling when running at high speed, since small errors can lead to catastrophic failures such as crashes [152, 177]. There are works that attempt to mitigate this issue by applying neural system identification [68, 166, 177] to adapt the model to different environmental factors such as tire degradation, ground surface imperfections [68], and towed objects [177]. Nevertheless, these methods still require assumptions about the specific car setup (e.g., size and wheelbase of the car).

A key question that arises is: *Can we train a generalist wheeled-robot dynamics model that achieves the performance of a specialist model for each setup?* In this work, we propose AnyCar (depicted in Figure 4.2), an initial effort to train a vehicle dynamics transformer that can predict the trajectory of various cars in various settings through in-context adaptation. Our contributions are three-fold:

- We build a universal synthetic data generator that collects data across diverse vehicles and environments, using physics engines with varying levels of fidelity (e.g., DBM, MuJoCo, Isaac Sim, Assetto Corsa Gym).
- We propose a two-phase robust vehicle dynamics transformer training method that leverages simulation pre-training and real-world fine-tuning to handle sim2real mismatches and state-estimation errors.

- We integrate the dynamics transformer with a sampling-based MPC and demonstrate real-world performance on different car platforms and in different environments, both indoor and outdoor, achieving up to 54% performance improvement over the baseline methods.

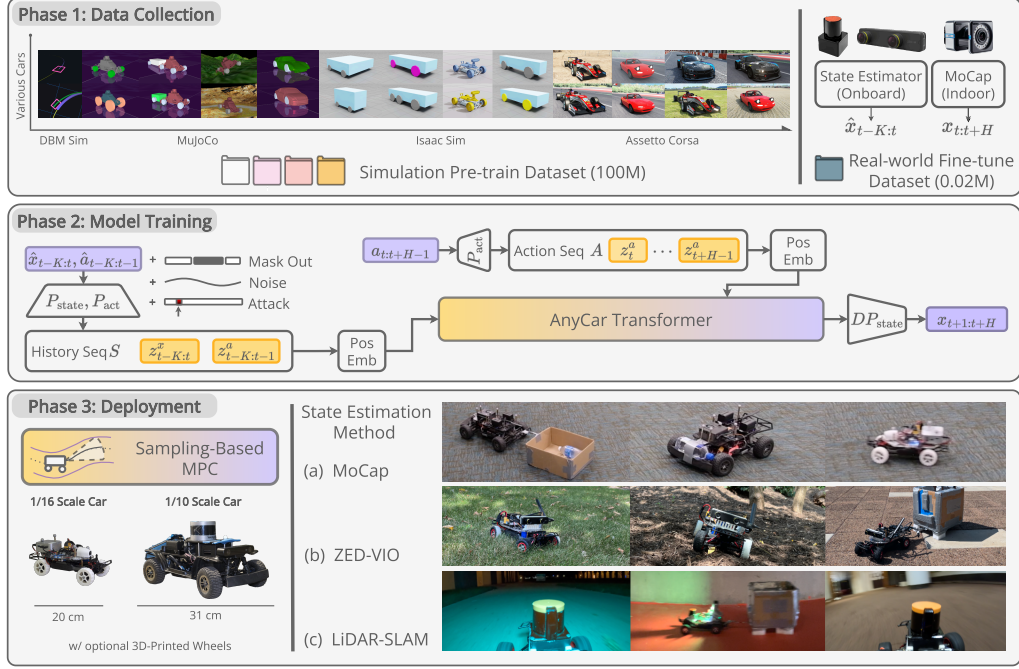


Figure 4.2: AnyCar System Pipeline. **Phase 1:** We collect 100M data in 4 different simulations for pre-training and 0.02M few-shot real-world data for fine-tuning the model. **Phase 2:** We pre-train the model with the simulation dataset and enhance prediction robustness through masking, adding noise, and attacking the inputs. We also fine-tune using the fine-tuning dataset. **Phase 3:** We deploy AnyCar in the wild under state estimation error (using SLAM [31, 104, 105] and VIO [153]) to control different vehicles (1/10 scale, 1/16 scale) with different settings (tow object, 3D-printed wheels) on different terrains.

4.2 Overview

Notation. x_t, a_t are state and action at time step t . We use $x_{1:t}$ to denote a sequence $\{x_1, \dots, x_t\}$. $\hat{x}$ denotes estimation of x (e.g., from VIO) and $\hat{a}$ is a with noise.

To demonstrate the practical application of our method, we focus on trajectory tracking

in unstructured environments for a range of vehicles, which can be formulated as follows:

$$\begin{aligned} & \underset{a_{0:T}}{\text{maximize}} && \sum_{t=0}^T R(x_t, a_t) \\ & \text{subject to} && x_{t+1} = f(x_t, a_t, c_t), \quad \forall t = 0, 1, \dots, T \end{aligned}$$

where $R(x_t, a_t)$ is the reward function and $x_{t+1} = f(x_t, a_t, c_t)$ represents the system dynamics with c_t representing all physics characteristics related to car dynamics and the environment conditions such as terrain, payload, .etc. The state is defined by $x_t \triangleq [p_t^x, p_t^y, \psi_t, \dot{p}_t^x, \dot{p}_t^y, \omega]$, which contains position (p_t^x, p_t^y) , heading angle ψ_t , linear velocity $(\dot{p}_t^x, \dot{p}_t^y)$, and angular velocity ω . The action $a_t \triangleq [\mathcal{T}, \delta]$ contains throttle $\mathcal{T}$ and steering angle δ . Our model is a seq2seq model that can predict the future state sequence from *imperfect* state and action history sequence and future action sequence:

$$x_{t+1:t+H} \approx f_{\theta}^{\text{AnyCar}} \left(\underbrace{\hat{x}_{t-K:t}, \hat{a}_{t-K:t-1}}_{\text{noisy state and action history}}, \underbrace{a_{t:t+H-1}}_{\text{future actions}} \right), \quad (4.1)$$

where K denotes history length and H denotes prediction horizon (illustrated in [Figure 4.2](#) Phase 2). In our formulation, c_t is implicitly inferred from the state and action history and is not explicitly passed into the network. We train a transformer to approximate [Equation \(4.1\)](#) for various cars and terrains via its in-context adaptation capability. Our model not only can learn adaptive dynamics but also a filter that can handle noisy state estimation $\hat{x}$ and action $\hat{a}$. Compared with previous works that assumes one specific car model with limited adaptable parameters [[68](#), [166](#), [171](#), [177](#), [181](#)], AnyCar can adapt to various types of car with or without assumptions about the environment. To learn θ , we design two-stage training pipeline. In the first stage, in [Section 4.3](#), we generate large scale dataset which contains trajectories of various cars in different terrains using different physics simulations; we then pre-train the AnyCar transformer $f_{\theta_{\text{sim}}}^{\text{AnyCar}}$ ([Figure 4.2](#) Phase 2) in [Section 4.3.2](#). In the second stage, we collect few-shot real-world data and fine-tune the transformer $f_{\theta_{\text{fine-tune}}}^{\text{AnyCar}}$ in [Section 4.3.3](#). After these two stages, the model $f_{\theta_{\text{fine-tune}}}^{\text{AnyCar}}$ can accurately predict future trajectory for different cars even under state-estimation error.

With an accurate dynamics model, we apply the Model Predictive Path Integral (MPPI) to perform trajectory tracking. MPPI is a sampling-based MPC approach that minimizes the cost-to-go for sampled trajectories and selects optimal controls by weighting multiple

candidate trajectories based on their performance. We choose MPPI for its ability to leverage parallel computation and its training-free nature, with no need for a reduced-order model. In [Section 4.4.1](#), we describe the detailed system design for trajectory tracking using AnyCar transformer with MPPI to achieve control at 50Hz in the real world. Finally, in [Section 4.6](#), we showcase the deployment of AnyCar in various real-world scenarios, demonstrating its versatility and robustness across different vehicles and environments

4.3 Model and Data

In this section, we describe the dataset collection and model training strategies of AnyCar. As shown in [Figure 4.2](#) Phase 1 and Phase 2, we highlight our data collection in massive simulation and few-shot real-world data, and our robust training and finetuning pipeline.

4.3.1 Pre-Training with Massive Simulated Data

Scene Generation. To collect a diverse dataset, we leverage the low-cost nature of simulation and generate a large amount of simulated data. Our data generation has three sources of diversity: 1) dynamics, 2) scenario, 3) controller. As illustrated in [Figure 4.2](#) Phase 1, we synthesize trajectories of various cars and terrains via four distinct simulators: Dynamic Bicycle Model [68] (DBM)-based numeric simulation, MuJoCo, IsaacSim, and Assetto Corsa Gym [134].

Curriculum Model Training. To ensure data distribution coverage, we diversify between on-policy and off-policy data via a two-stage curriculum learning method. In stage one, we collect high-volume off-policy data as warm-up, by building a hybrid controller where we use a pure pursuit controller for steering δ and a PD controller for throttle $\mathcal{T}$, to track randomly synthesized reference tracks. After collecting 200M timesteps that are usable for training a general model for non-agile tasks (in which the target velocity is smaller than the physical limit), we switch to stage two, where we deploy an on-policy NN-MPPI controller (described in [Section 4.4.1](#)) to track agile trajectories, and periodically update the network with the collected on-policy trajectory.

4.3.2 Robust In-Context Adaptive Dynamics Model

Model Structure. As illustrated in Figure 4.2 Phase 2, the historical states in Equation (4.1) are linearly projected into a 64-dimensional latent space with an encoder layer $P_{\text{state}}(x) : \mathbb{R}^6 \rightarrow \mathbb{R}^{64}$. The historical actions pass through a different encoder $P_{\text{act}}(a) : \mathbb{R}^2 \rightarrow \mathbb{R}^{64}$ of similar size. We then interleave and stack them to make a complete history token sequence $S^{(2K-1) \times 64}$, which is supplied to the transformer as the context. The future actions are tokenized with $P_{\text{act}}(a)$ and stacked as a sequence $A^{H \times 64}$. Both S and A sequences are then summed with two learnable positional encoders. After passing the context S and input A to the transformer decoder, we obtain $S'^{H \times 64}$, which is then down-projected into the state-space with $DP_{\text{state}}(x) : \mathbb{R}^{64} \rightarrow \mathbb{R}^6$. This becomes the final state sequence prediction.

Robust Training. To learn a robust model f_θ for Equation (4.1) under various disturbances, we propose to add three techniques in pre-training, i.e., mask out, add noise, and attack (Figure 4.2). We implement *mask out* by applying randomized cross-attention mask to transformer, then *add noise* $\epsilon \sim \mathcal{N}(0, \epsilon^{\max})$. We also add *attack*: unreasonably large or small values to random dimensions in history to simulate state estimation errors in the real-world. As discussed in Section 4.3, these techniques collaboratively improves robustness for successful real-world deployment.

4.3.3 Fine-Tuning with state-estimation error reduction

After pre-training $f_{\theta_{\text{sim}}}^{\text{AnyCar}}$, we fine-tune the model with 10 minutes of real-world data (0.02M) to reduce the sim2real gap and mitigate state-estimation error. To collect real-world data, we control the car via joystick to follow a few random curved trajectories in a motion capture field and collect both the state from on-board state-estimators and the ground truth states provided by the Vicon system, as shown in Figure 4.2 Phase 1. We then fine-tune $f_{\theta_{\text{sim}}}^{\text{AnyCar}}$ on collected data under the constraint of $\|\theta_{\text{fine-tune}} - \theta_{\text{sim}}\|_2 \leq \epsilon_{\text{tune}}$ to prevent catastrophic forgetting. In practice, we also apply data rehearsal to assist. The model after fine-tuning $f_{\theta_{\text{fine-tune}}}^{\text{AnyCar}}$ is ready to deploy at zero-shot.

4.4 System Design

In this section, we describe the system that leverages AnyCar to perform various trajectory tracking tasks.

4.4.1 MPPI Controller

As introduced in Section 4.2, we choose to evaluate our method with MPPI instead of RL and MPC, because RL would have required optimizing additional policy and value function parameters, which is outside the scope of this work. Likewise, neural MPC [141] would have resulted in a reduced-order approximation of our neural dynamics model that is undesired for a fair evaluation.

We employ a sample-efficient MPPI implementation in [184], which maintains an adaptive sampling covariance to achieve an optimal convergence rate on the cost function.

Trajectory Sampling

Let x_t, a_t be state and action at time t . For given control horizon H , we randomly sample N action sequences $\{a_{t:t+H-1}^i\}_i^N$ in a normal distribution, whose mean and covariance are computed with [184]. We roll out each action sequence with the AnyCar model and compute their cumulative rewards. In order to generate smooth action sequences, we re-parameterize the control sequence $a_{0:T}$ with a set of time-indexed knots represented by $\theta_{0:k}$ [55]. Given query point τ , the control can be evaluated by $a_\tau = \text{spline}(\tau; (\tau_{0:k}, \theta_{0:k}))$.

Reward function

To motivate tracking reference waypoints, we use the single-step reward function from [42], which is given by

$$r(x, a, \hat{x}) = w_1 \|p - \hat{p}\|^2 + w_2 \|\psi - \hat{\psi}\| + w_3 \|v_x - \hat{v}_x\| + w_4 \|\delta a\|,$$

where p denotes position, ψ is heading angle, v_x is longitude velocity, and δa is action increment.

4. Learning Universal Dynamics Model for Agile and Adaptive Mobility

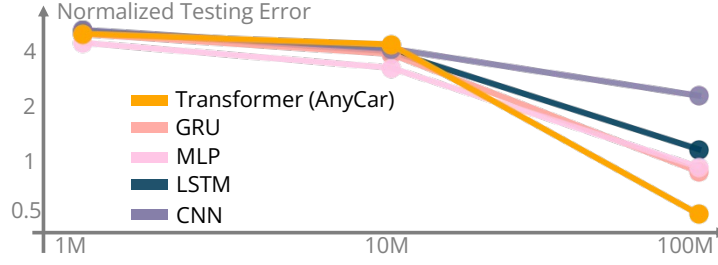


Figure 4.3: Comparison of different model structures and data scales. The reported testing error is normalized using the mean and standard deviation of the evaluation dataset.

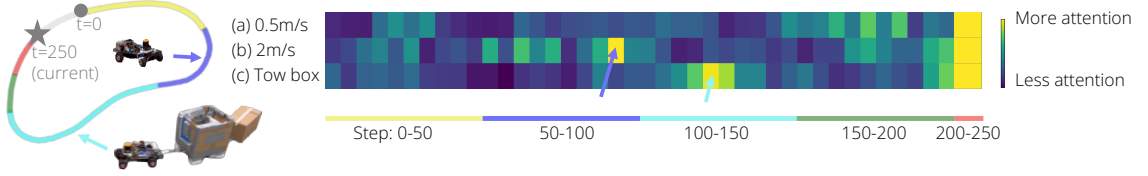


Figure 4.4: Visualization of AnyCar’s transformer attention in three real-world settings: (a) low speed at 0.5 m/s, (b) high speed at 2 m/s, and (c) towing an object at 2 m/s, all tracking the same reference trajectory. AnyCar’s transformer consistently focuses on the nearest 50 steps across all settings and adaptively attends to different sections of the track. For example, it attends to the first corner in setting (b) and the second corner in setting (c).

Computation

We implement the MPPI in JAX [8] and the transformer model in TransformerEngine. We evaluate 600 action sequence samples on the model in parallel, achieving 20 ms (50 Hz) real-time performance on a RTX 4090 GPU. We note that further optimizations, such as KV caching, sharing history token attentions between samples, and TensorRT conversion can enable edge deployment.

4.4.2 State Estimation

In our experiments with ground truth data, we use a motion capture system to directly observe the position and velocity of the vehicle. For in-the-wild experiments, we estimate the linear and angular velocity of the vehicle by fusing motor odometry and IMU data using an Extended Kalman Filter (EKF) [114]. We then correct for the odometry drifting using 2D LiDAR SLAM [31, 104, 105], or 3D SLAM with VIO. We will demonstrate AnyCar’s fine-tuning pipeline can adapt to state estimation error in [Section 4.6](#).

Table 4.1: Indoor Results using Ground Truth State-Estimation (Motion Capture)

Setting	Method	$E^{\text{Prediction}} \downarrow$	$E^{\text{Tracking}} \downarrow$
Few-shot performance in fine-tuned scenarios			
1/10 Scale Car	PP	-	0.45 ± 0.02
+ Tow Box	AnyCar (Ours)	0.27 ± 0.24	0.35 ± 0.04
	AnyCar w/o FT	0.54 ± 0.55	0.47 ± 0.23
	Specialist	0.14 ± 0.09	0.43 ± 0.05
1/10 Scale Car	PP	-	0.54 ± 0.03
+ Payloads	AnyCar (Ours)	0.11 ± 0.08	0.41 ± 0.03
	AnyCar w/o FT	0.21 ± 0.11	0.47 ± 0.17
	Specialist	0.26 ± 0.11	0.51 ± 0.03
Zero-shot generalization in unseen scenarios			
1/10 Scale Car	PP	-	0.79 ± 0.62
+ Plastic wheels	AnyCar (Ours)	0.26 ± 0.32	0.335 ± 0.03
(All 4 wheels)	AnyCar w/o FT	0.32 ± 0.21	0.45 ± 0.15
	Specialist	0.35 ± 0.15	0.334 ± 0.06
1/10 Scale Car	PP	-	0.52 ± 0.05
+ Plastic wheels	AnyCar (Ours)	0.12 ± 0.08	0.39 ± 0.05
(Front 2 wheels)	AnyCar w/o FT	0.20 ± 0.11	0.49 ± 0.09
	Specialist	0.27 ± 0.11	0.41 ± 0.04
1/10 Scale Car	PP	-	0.57 ± 0.03
+ Plastic wheels	AnyCar (Ours)	0.09 ± 0.06	0.49 ± 0.09
+ Tow box	AnyCar w/o FT	0.18 ± 0.11	0.52 ± 0.09
	Specialist	0.14 ± 0.05	0.60 ± 0.04
1/16 Scale Car	PP	-	0.37 ± 0.16
+ Plastic wheels	AnyCar (Ours)	0.17 ± 0.08	0.31 ± 0.06
	AnyCar w/o FT	0.25 ± 0.14	0.44 ± 0.08
	Specialist	0.26 ± 0.11	0.55 ± 0.07

4.4.3 Low-Level Controller

Upon receiving a throttle (acceleration) and steering (angle) command, a low-level module maps the throttle to motor current with a simple linear mapping, and maps the steering angle to servo angle. Adapting to the discrepancies in these rough mappings and the latency in actuator responses is also a goal of our fine-tuning pipeline.

4.5 Experiments and Results

4.6 Experiment

In this section, we aim to demonstrate the capability of the proposed AnyCar by addressing the following questions:

- **Q1:** Can our model generalize to various cars and terrains, and outperform specialist models?
- **Q2:** Can our model maintain its adaptation capability even with imperfect state estimation?
- **Q3:** Why does the proposed robust vehicle dynamics transformer outperform other baseline models?

Baselines. We compare AnyCar with three baseline methods: 1) *AnyCar w/o FT*: AnyCar without the real-world fine-tuning phase, 2) *PP*: Pure Pursuit controller for steering and PID controller for velocity tracking, and 3) *Specialist*: a DBM model with system identification. We also consider two types of state estimator setups: 1) motion capture (indoor), which can be treated as ground truth, and 2) SLAM [31, 104, 105] (indoor or outdoor), which is less accurate than motion capture and much prone to drifting issues.

Metrics. We use the model prediction error $E^{\text{Prediction}} \triangleq \|x_{t+1:t+H}^{\text{pred}} - x_{t+1:t+H}^{\text{gt}}\|_2$ to assess prediction accuracy. We also define $E^{\text{Tracking}} \triangleq w_2 \|p_t - \hat{p}_t\|_2 + w_3 \|v_t - \hat{v}_t\|_2$, where w_2 and w_3 are the same weights defined for position and velocity tracking rewards in Section 4.4.1, representing the weighted sum of lateral error and velocity tracking error, to evaluate trajectory tracking performance.

4.6.1 Evaluate Model In-context Adaptation Capability

To answer **Q1**, we use motion capture to provide ground truth for state estimation in the real world. An 1/10 Scale Car was employed to tow objects and carry payloads, to create a fine-tuning dataset collected by human teleoperation. This dataset was then used to fine-tune our dynamics model. The model was evaluated across two categories: **few-shot performance** (e.g., towing objects and varying payloads, both present in the fine-tuning dataset) and **zero-shot generalization** (e.g., changing 3D-printed wheels, towing objects

4. Learning Universal Dynamics Model for Agile and Adaptive Mobility

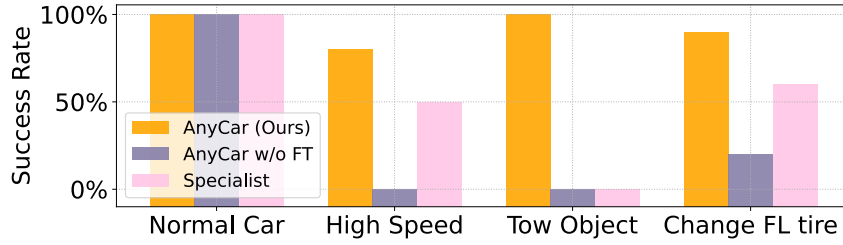


Figure 4.5: Comparison with baselines in the wild.

Table 4.2: Indoor Results with State-Estimation Error (VIO)

Setting	Method	$E^{\text{Prediction}} \downarrow$	$E^{\text{Tracking}} \downarrow$
Few-shot performance in fine-tuned scenarios			
1/16 Scale Car + Low Speed	AnyCar (Ours)	0.30 ± 0.06	0.35 ± 0.02
	AnyCar w/o FT	0.32 ± 0.07	0.48 ± 0.06
	Specialist	0.33 ± 0.05	0.42 ± 0.01
Zero-shot generalization in unseen scenarios			
1/16 Scale Car + High Speed	AnyCar (Ours)	0.44 ± 0.11	0.57 ± 0.03
	AnyCar w/o FT	0.52 ± 0.15	1.30 ± 0.11
	Specialist	0.48 ± 0.10	1.26 ± 0.89
1/16 Scale Car + Tow 2 Box	AnyCar (Ours)	0.34 ± 0.10	0.57 ± 0.02
	AnyCar w/o FT	0.41 ± 0.14	1.41 ± 0.09
	Specialist	0.39 ± 0.09	0.93 ± 0.62

with modified wheels, and using a smaller car model with altered wheels). We use a trajectory optimization method [181] to compute an on-the-edge reference trajectory on a raceline. Our method, along with other baselines (AnyCar w/o FT, PP, Specialist), was then deployed to track this trajectory. The evaluation considered both model prediction error and closed-loop tracking error with the full controller. Results in Table 4.1 show AnyCar reaching and outperforming baselines in terms of prediction error and tracking error in both few-shot and zero-shot cases. Perhaps more exciting, we observe similar benefits seen in cross-embodiment works like RT-X [119], where fine-tuning the model with the real-world data of one robot (1/10 scale car) leads to a performance boost on another robot (1/16 scale car) even though the dynamics and trajectory tracked may be completely different.

4.6.2 Evaluate Model Capability in the Wild

To address **Q2**, we set up a 2D LiDAR-based SLAM stack on an 1/16 scale car. The car was teleoperated to follow an S-shaped path at low speed, collecting odometry data from both the SLAM system and motion capture to create a small-scale dataset of 24,000 timesteps. The model was then fine-tuned using the method in [Section 4.3.3](#). Using the fine-tuned model (AnyCar), we first evaluated our method in an indoor environment under different conditions (low speed, high speed, towing two boxes) with SLAM, and computed metrics using ground truth data. The results in [Table 4.2](#) show AnyCar outperforms baselines with peak improvement of 54%. Based on [Table 4.1](#) and [Table 4.2](#), we prove the necessity of fine-tuning that aligning the dynamics model to handle state estimation error.

The system is then moved to an outdoor environment without further modification. To validate the robustness to state-estimation capabilities of AnyCar, we set up a narrow corridor along the reference trajectory, allowing the car to pass with a 10 cm tolerance (demonstrated in [Figure 4.1](#)). Poor tracking performance would result in the car colliding with the walls. We compared our method with AnyCar w/o FT and Specialist by calculating the percentage of times the car successfully passed all checkpoints without collision (success rate) for each method. The results in [Figure 4.5](#) show AnyCar achieves the highest success rate in all settings, with the specialist failing consistently due to the state estimation error. We also visualize the transformer attention across different settings in [Figure 4.4](#), which demonstrates AnyCar’s in-context adaptation in various settings. In addition to SLAM-based state-estimator, we also show the deployment with ZED-VIO [[153](#)] in supplementary videos.

4.6.3 Interpret the expressiveness and robustness of AnyCar

To interpret the success of AnyCar, we argue that scaling and robust training are integral parts of our method. We conducted experiments on datasets of varying sizes ranging from 1M (million) to 100M. Various model architectures, including Transformer, LSTM, GRU, CNN, and MLP, are evaluated, and the normalized prediction error is calculated for each. To ensure the models could be used with MPPI for 50Hz control, we limit the model size to 200K parameters. We create another 1M testing dataset to evaluate all trained models. The results shown in [Figure 4.3](#) demonstrate that as the training dataset size increased from 10M to 100M timesteps, the prediction errors decreased significantly, with the transformer

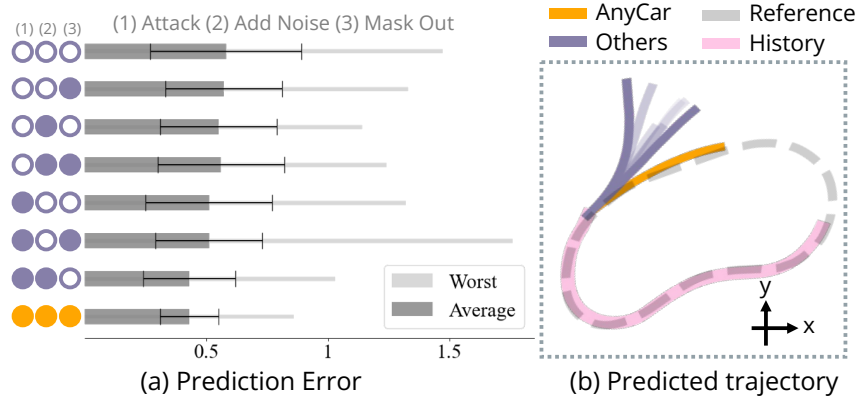


Figure 4.6: Comparison of AnyCar robust training methods. (a) Evaluate prediction error in real-world trajectories. (b) Demonstrate predicted trajectory of different methods.

model performing best at the 100M scale. This highlights that the transformer structure is the most effective for modeling diverse car dynamics and environments compared with baseline models. However, scaling the data and using the appropriate model structure alone are insufficient. Under the optimal data scale and model configuration, we found it crucial to apply the proposed robust training methods (including attack, noise, and mask-out strategies), as discussed in [Section 4.3.2](#). We systematically evaluate all combinations of these components, resulting in a total of 8 pre-trained models. Each model is fine-tuned using the same dataset and evaluated on real trajectory. The results, shown in [Figure 4.6\(a\)](#), demonstrate that the model achieves the highest prediction accuracy and stability only when all robust training components are activated. For instance, [Figure 4.6\(b\)](#) shows that without full robust training, the transformer’s predictions are vulnerable to noisy state estimation, leading to significant errors. The combination of model selection, large-scale data, and robust training explains why the AnyCar performs better than the baseline models.

4. Learning Universal Dynamics Model for Agile and Adaptive Mobility

Bibliography

- [1] Minttu Alakuijala, Gabriel Dulac-Arnold, Julien Mairal, Jean Ponce, and Cordelia Schmid. Residual reinforcement learning from demonstrations. *arXiv preprint arXiv:2106.08050*, 2021. 3.2.3
- [2] Chae H An, Christopher G Atkeson, and John M Hollerbach. Estimation of inertial parameters of rigid body links of manipulators. In *1985 24th IEEE Conference on Decision and Control*, pages 990–995. IEEE, 1985. 3.2.2
- [3] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement–residual rl for precise assembly. *arXiv preprint arXiv:2407.16677*, 2024. 3.2.3
- [4] Rika Antonova, Jingyun Yang, Priya Sundaesan, Dieter Fox, Fabio Ramos, and Jeanette Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3):5819–5826, 2022. 3.2.2
- [5] Karl Johan Åström and Peter Eykhoff. System identification—a survey. *Automatica*, 7(2):123–162, 1971. 3.2.2
- [6] Genesis Authors. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL <https://github.com/Genesis-Embodied-AI/Genesis>. 3.5
- [7] Boston Dynamics. Atlas — partners in parkour. <https://www.youtube.com/watch?v=tF4DML7FIWk>, 2021. 2.1
- [8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake Vander{P}las, Skye Wanderman-{M}ilne, and Qiao Zhang. google/jax, September 2024. URL <https://github.com/google/jax>. original-date: 2018-10-25T21:25:02Z. 4.4.1
- [9] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manju-

- nath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023. ISBN 978-0-9923747-9-2. doi: 10.15607/RSS.2023.XIX.025. URL <http://www.roboticsproceedings.org/rss19/p025.pdf>. 4.1
- [10] Rodney Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986. 2.4
- [11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL <https://arxiv.org/abs/2005.14165>. 1
- [12] Luigi Campanaro, Siddhant Gangapurwala, Wolfgang Merkt, and Ioannis Havoutis. Learning and deploying robust locomotion policies with minimal dynamics randomization. In *6th Annual Learning for Dynamics & Control Conference*, pages 578–590. PMLR, 2024. 3.6.3
- [13] Joao Carvalho, Dorothea Koert, Marek Daniv, and Jan Peters. Residual robot learning for object-centric probabilistic movement primitives. *arXiv preprint arXiv:2203.03918*, 2022. 3.2.3
- [14] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan Ratliff, and Dieter Fox. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8973–8979. IEEE, 2019. 3.2.2
- [15] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: reinforcement learning via sequence modeling. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS ’21, pages 15084–15097, Red Hook, NY, USA, June 2024. Curran Associates Inc. ISBN 978-1-71384-539-3. 4.1
- [16] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024. 2.1, 2.1, 2.2.2, ??, 2.2.3, 2.3, 2.4, 3.1, 3.2.1

- [17] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024. 1, 2.1, 2.4
- [18] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-TeleVision: Teleoperation with Immersive Active Visual Feedback, July 2024. URL <http://arxiv.org/abs/2407.01512>. arXiv:2407.01512 [cs]. 4.1
- [19] Nuttapong Chentanez, Matthias Müller, Miles Macklin, Viktor Makoviychuk, and Stefan Jeschke. Physics-based motion capture imitation with deep reinforcement learning. *Proceedings - MIG 2018: ACM SIGGRAPH Conference on Motion, Interaction, and Games*, 2018. 2.4
- [20] Cheng Chi, Benjamin Burchfiel, Eric Cousineau, Siyuan Feng, and Shuran Song. Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4):389–404, 2024. 3.2.3
- [21] Yufeng Chi, Qiayuan Liao, Junfeng Long, Xiaoyu Huang, Sophia Shao, Borivoje Nikolic, Zhongyu Li, and Koushil Sreenath. Demonstrating berkeley humanoid lite: An open-source, accessible, and customizable 3d-printed humanoid robot, 2025. URL <https://arxiv.org/abs/2504.17249>. 1
- [22] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The mit humanoid robot: Design, motion planning, and control for acrobatic behaviors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2021. 2.4
- [23] Xiangxiang Chu, Limeng Qiao, Xinyu Zhang, Shuang Xu, Fei Wei, Yang Yang, Xiaofei Sun, Yiming Hu, Xinyang Lin, Bo Zhang, and Chunhua Shen. Mobilevlm v2: Faster and stronger baseline for vision language model, 2024. URL <https://arxiv.org/abs/2402.03766>. 1
- [24] Stefano Dafarra, Ugo Pattacini, Giulio Romualdi, Lorenzo Rapetti, Riccardo Grieco, Kourosh Darvish, Gianluca Milani, Enrico Valli, Ines Sorrentino, Paolo Maria Viceconte, et al. icub3 avatar system: Enabling remote fully immersive embodiment of humanoid robots. *Science Robotics*, 9(86):eadh3834, 2024. 2.4
- [25] Kourosh Darvish, Luigi Penco, Joao Ramos, Rafael Cisneros, Jerry Pratt, Eiichi Yoshida, Serena Ivaldi, and Daniele Pucci. Teleoperation of humanoid robots: A survey. *IEEE Transactions on Robotics*, 39(3):1706–1727, 2023. 2.4
- [26] Todor Davchev, Kevin Sebastian Luck, Michael Burke, Franziska Meier, Stefan Schaal, and Subramanian Ramamoorthy. Residual learning from demonstration: Adapting dmps for contact-rich manipulation. *IEEE Robotics and Automation Letters*, 7(2):4488–4495, 2022. 3.2.3
- [27] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai

- Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanbiao Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yudian Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL <https://arxiv.org/abs/2412.19437>. 1
- [28] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. October 2020. URL <https://openreview.net/forum?id=YicbFdNTTy>. 4.1
- [29] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1290–1296. IEEE, 2021. 3.2.2
- [30] Pranay Dugar, Aayam Shrestha, Fangzhou Yu, Bart van Marum, and Alan Fern.

- Learning multi-modal whole-body control for real-world humanoid robots. *arXiv preprint arXiv:2408.07295*, 2024. ??, 2.4
- [31] Dieter Fox. KLD-Sampling: Adaptive Particle Filters. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL <https://proceedings.neurips.cc/paper/2001/hash/c5b2cebf15b205503560c4e8e6d1ea78-Abstract.html>. (document), 4.2, 4.4.2, 4.6
- [32] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. HumanPlus: Humanoid Shadowing and Imitation from Humans. 4.1
- [33] 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. 3.2.2
- [34] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*, 2024. 2.1, 2.1, 2.2.2, ??, 2.2.3, 2.3, 2.4, 3.2.1
- [35] Feng Gao, Chao Yu, Yu Wang, and Yi Wu. Neural internal model control: Learning a robust control policy via predictive error feedback. *arXiv preprint arXiv:2411.13079*, 2024. 3.2.2
- [36] Jiawei Gao, Ziqin Wang, Zeqi Xiao, Jingbo Wang, Tai Wang, Jinkun Cao, Xiaolin Hu, Si Liu, Jifeng Dai, and Jiangmiao Pang. Coohoi: Learning cooperative human-object interaction with manipulated object dynamics. *arXiv preprint arXiv:2406.14558*, 2024. 3.2.1
- [37] Maxime Gautier, Pierre-Olivier Vandanjon, and Alexandre Janot. Dynamic identification of a 6 dof robot without joint position data. In *2011 IEEE International Conference on Robotics and Automation*, pages 234–239. IEEE, 2011. 3.2.2
- [38] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jérôme Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic design of nao humanoid. In *2009 IEEE international conference on robotics and automation*, pages 769–774. IEEE, 2009. 2.4
- [39] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning. *arXiv preprint arXiv:2408.14472*, 2024. 3.1, 3.2.1, 3.2.2
- [40] Zhaoyuan Gu, Junheng Li, Wenlan Shen, Wenhao Yu, Zhaoming Xie, Stephen McCrory, Xianyi Cheng, Abdulaziz Shamsah, Robert Griffin, C. Karen Liu, Abderrahmane Kheddar, Xue Bin Peng, Yuke Zhu, Guanya Shi, Quan Nguyen, Gordon Cheng, Huijun Gao, and Ye Zhao. Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning, 2025. URL <https://arxiv.org/abs/2501.02116>. 1

- [41] Siddhant Halder, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023. 3.2.3
- [42] Tyler Han, Alex Liu, Anqi Li, Alexander Spitzer, Guanya Shi, and Byron Boots. Model Predictive Control for Aggressive Driving Over Uneven Terrain. 4.1, 4.4.1
- [43] Yong Han, Jianhua Wu, Chao Liu, and Zhenhua Xiong. An iterative approach for accurate dynamic model identification of industrial robots. *IEEE Transactions on Robotics*, 36(5):1577–1594, 2020. 3.2.2
- [44] Mohamed Hassan, Yunrong Guo, Tingwu Wang, Michael Black, Sanja Fidler, and Xue Bin Peng. Synthesizing physical character-scene interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023. 3.2.1
- [45] Kris Hauser, Eleanor ‘Nell’ Watson, Joonbum Bae, Josh Bankston, Sven Behnke, Bill Borgia, Manuel G Catalano, Stefano Dafarra, Jan BF van Erp, Thomas Ferris, et al. Analysis and perspectives on the ana avatar xprize competition. *International Journal of Social Robotics*, pages 1–32, 2024. 2.4
- [46] Guanqi He, Yogita Choudhary, and Guanya Shi. Self-supervised meta-learning for all-layer dnn-based adaptive control with stability guarantees. *arXiv preprint arXiv:2410.07575*, 2024. 3.2.3
- [47] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. OmniH2O: Universal and Dexterous Human-to- Humanoid Whole-Body Teleoperation and Learning. . 4.1
- [48] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile But Safe: Learning Collision-Free High-Speed Legged Locomotion. . 4.1
- [49] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. *arXiv preprint arXiv:2406.08858*, 2024. 1, 2, 2.1, 2.1, 2.2.2, ??, 2.2.3, 2.2.4, 2.2.4, 2.2.5, 2.3, 2.4, 3.1, 3.2.1, 3.3.2
- [50] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024. 2.1, 2.1, 2.2.2, ??, 2.2.3, 2.3, 2.4, 3.1, 3.2.1, 3.3.1, 3.3.1, 3.3.2
- [51] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation, March 2024. URL <http://arxiv.org/abs/2403.04436>. arXiv:2403.04436 [cs, eess]. 4.1
- [52] Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, et al. Hover: Versatile neural

- whole-body controller for humanoid robots. *arXiv preprint arXiv:2410.21229*, 2024. [2](#), [3.1](#)
- [53] Tairan He, Chong Zhang, Wenli Xiao, Guanqi He, Changliu Liu, and Guanya Shi. Agile but safe: Learning collision-free high-speed legged locomotion. *arXiv preprint arXiv:2401.17583*, 2024. [2.2.2](#)
- [54] Eric Heiden, Christopher E Denniston, David Millard, Fabio Ramos, and Gaurav S Sukhatme. Probabilistic inference of simulation parameters via parallel differentiable simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 3638–3645. IEEE, 2022. [3.2.2](#)
- [55] Taylor Howell, Nimrod Gileadi, Saran Tunyasuvunakool, Kevin Zakka, Tom Erez, and Yuval Tassa. Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo, December 2022. URL <http://arxiv.org/abs/2212.00541>. arXiv:2212.00541 [cs, eess]. [4.4.1](#)
- [56] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Hao-Shu Fang, Shibo Zhao, Shayegan Omidshafiei, Dong-Ki Kim, Ali akbar Agha-mohammadi, Katia Sycara, Matthew Johnson-Roberson, Dhruv Batra, Xiaolong Wang, Sebastian Scherer, Chen Wang, Zsolt Kira, Fei Xia, and Yonatan Bisk. Toward general-purpose robots via foundation models: A survey and meta-analysis, 2024. URL <https://arxiv.org/abs/2312.08782>. [1](#)
- [57] Kevin Huang, Rwik Rana, Alexander Spitzer, Guanya Shi, and Byron Boots. Datt: Deep adaptive trajectory tracking for quadrotor control. *arXiv preprint arXiv:2310.09053*, 2023. [3.2.2](#)
- [58] Tao Huang, Junli Ren, Huayi Wang, Zirui Wang, Qingwei Ben, Muning Wen, Xiao Chen, Jianan Li, and Jiangmiao Pang. Learning humanoid standing-up control across diverse postures, 2025. URL <https://arxiv.org/abs/2502.08378>. [1](#)
- [59] 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*, 4(26):eaau5872, 2019. [1](#), [3.1](#), [3.2.2](#)
- [60] Imagen-Team-Google, :, Jason Baldridge, Jakob Bauer, Mukul Bhutani, Nicole Brichtova, Andrew Bunner, Lluís Castrejon, Kelvin Chan, Yichang Chen, Sander Dieleman, Yuqing Du, Zach Eaton-Rosen, Hongliang Fei, Nando de Freitas, Yilin Gao, Evgeny Gladchenko, Sergio Gómez Colmenarejo, Mandy Guo, Alex Haig, Will Hawkins, Hexiang Hu, Huilian Huang, Tobenna Peter Igwe, Christos Kaplanis, Siavash Khodadadeh, Yelin Kim, Ksenia Konyushkova, Karol Langner, Eric Lau, Rory Lawton, Shixin Luo, Soňa Mokrá, Henna Nandwani, Yasumasa Onoe, Aäron van den Oord, Zarana Parekh, Jordi Pont-Tuset, Hang Qi, Rui Qian, Deepak Ramachandran, Poorva Rane, Abdullah Rashwan, Ali Razavi, Robert Ri-

achi, Hansa Srinivasan, Srivatsan Srinivasan, Robin Strudel, Benigno Uribe, Oliver Wang, Su Wang, Austin Waters, Chris Wolff, Auriel Wright, Zhisheng Xiao, Hao Xiong, Keyang Xu, Marc van Zee, Junlin Zhang, Katie Zhang, Wenlei Zhou, Konrad Zolna, Ola Aboubakar, Canfer Akbulut, Oscar Akerlund, Isabela Albuquerque, Nina Anderson, Marco Andreetto, Lora Aroyo, Ben Bariach, David Barker, Sherry Ben, Dana Berman, Courtney Biles, Irina Blok, Pankil Botadra, Jenny Brennan, Karla Brown, John Buckley, Rudy Bunel, Elie Bursztein, Christina Butterfield, Ben Caine, Viral Carpenter, Norman Casagrande, Ming-Wei Chang, Solomon Chang, Shamik Chaudhuri, Tony Chen, John Choi, Dmitry Churbanau, Nathan Clement, Matan Cohen, Forrester Cole, Mikhail Dektiarev, Vincent Du, Praneet Dutta, Tom Eccles, Ndidi Elue, Ashley Feden, Shlomi Fruchter, Frankie Garcia, Roopal Garg, Weina Ge, Ahmed Ghazy, Bryant Gipson, Andrew Goodman, Dawid Górny, Sven Gowal, Khyatti Gupta, Yoni Halpern, Yena Han, Susan Hao, Jamie Hayes, Jonathan Heek, Amir Hertz, Ed Hirst, Emiel Hoogeboom, Tingbo Hou, Heidi Howard, Mohamed Ibrahim, Dirichi Ike-Njoku, Joana Iljazi, Vlad Ionescu, William Isaac, Reena Jana, Gemma Jennings, Donovan Jenson, Xuhui Jia, Kerry Jones, Xiaoen Ju, Ivana Kajic, Christos Kaplanis, Burcu Karagol Ayan, Jacob Kelly, Suraj Kothawade, Christina Kouridi, Ira Ktena, Jolanda Kumakaw, Dana Kurniawan, Dmitry Lagun, Lily Lavitas, Jason Lee, Tao Li, Marco Liang, Maggie Li-Calis, Yuchi Liu, Javier Lopez Alberca, Matthieu Kim Lorrain, Peggy Lu, Kristian Lum, Yukun Ma, Chase Malik, John Mellor, Thomas Mensink, Inbar Mosseri, Tom Murray, Aida Nematzadeh, Paul Nicholas, Signe Nørly, João Gabriel Oliveira, Guillermo Ortiz-Jimenez, Michela Paganini, Tom Le Paine, Roni Paiss, Alicia Parrish, Anne Peckham, Vikas Peswani, Igor Petrovski, Tobias Pfaff, Alex Pirozhenko, Ryan Poplin, Utsav Prabhu, Yuan Qi, Matthew Rahtz, Cyrus Rashtchian, Charvi Rastogi, Amit Raul, Ali Razavi, Sylvestre-Alvise Rebuffi, Susanna Ricco, Felix Riedel, Dirk Robinson, Pankaj Rohatgi, Bill Rosgen, Sarah Rumbley, Moonkyung Ryu, Anthony Salgado, Tim Salimans, Sahil Singla, Florian Schroff, Candice Schumann, Tanmay Shah, Eleni Shaw, Gregory Shaw, Brendan Shillingford, Kaushik Shivakumar, Dennis Shtatnov, Zach Singer, Evgeny Sluzhaev, Valerii Sokolov, Thibault Sottiaux, Florian Stimberg, Brad Stone, David Stutz, Yu-Chuan Su, Eric Tabellion, Shuai Tang, David Tao, Kurt Thomas, Gregory Thornton, Andeep Toor, Cristian Udrescu, Aayush Upadhyay, Cristina Vasconcelos, Alex Vasiloff, Andrey Voynov, Amanda Walker, Luyu Wang, Miaosen Wang, Simon Wang, Stanley Wang, Qifei Wang, Yuxiao Wang, Ágoston Weisz, Olivia Wiles, Chenxia Wu, Xingyu Federico Xu, Andrew Xue, Jianbo Yang, Luo Yu, Mete Yurtoglu, Ali Zand, Han Zhang, Jiageng Zhang, Catherine Zhao, Adilet Zhaxybay, Miao Zhou, Shengqi Zhu, Zhenkai Zhu, Dawn Bloxwich, Mahyar Bordbar, Luis C. Cobo, Eli Collins, Shengyang Dai, Tulsee Doshi, Anca Dragan, Douglas Eck, Demis Hassabis, Sissie Hsiao, Tom Hume, Koray Kavukcuoglu, Helen King, Jack Krawczyk, Yeqing Li, Kathy Meier-Hellstern, Andras Orban, Yury Pinsky, Amar Subramanya, Oriol Vinyals, Ting Yu, and Yori Zwols. Imagen 3, 2024. URL

- <https://arxiv.org/abs/2408.07009>. 1
- [61] Yasuhiro Ishiguro, Tasuku Makabe, Yuya Nagamatsu, Yuta Kojio, Kunio Kojima, Fumihito Sugai, Yohei Kakiuchi, Kei Okada, and Masayuki Inaba. Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis. *IEEE Robotics and Automation Letters*, 5(4):6419–6426, 2020. 2.4
 - [62] Michael Janner, Qiyang Li, and Sergey Levine. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, volume 34, pages 1273–1286. Curran Associates, Inc., 2021. URL <https://proceedings.neurips.cc/paper/2021/hash/099fe6b0b444c23836c4a5d07346082b-Abstract.html>. 4.1
 - [63] Alexandre Janot, Pierre-Olivier Vandanjon, and Maxime Gautier. A generic instrumental variable approach for industrial robot identification. *IEEE Transactions on Control Systems Technology*, 22(1):132–145, 2013. 3.2.2
 - [64] 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*, 7(2):4630–4637, 2022. 3.2.2
 - [65] Mazeyu Ji, Xuanbin Peng, Fangchen Liu, Jialong Li, Ge Yang, Xuxin Cheng, and Xiaolong Wang. Exbody2: Advanced expressive humanoid whole-body control. *arXiv preprint arXiv:2412.13196*, 2024. 3.1, 3.2.1
 - [66] Yunfan Jiang, Chen Wang, Ruohan Zhang, Jiajun Wu, and Li Fei-Fei. Transic: Sim-to-real policy transfer by learning from online correction. *arXiv preprint arXiv:2405.10315*, 2024. 3.2.3
 - [67] Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 international conference on robotics and automation (ICRA)*, pages 6023–6029. IEEE, 2019. 3.2.3
 - [68] Dvij Kalaria, Qin Lin, and John M. Dolan. Adaptive Planning and Control with Time-Varying Tire Models for Autonomous Racing Using Extreme Learning Machine. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10443–10449, May 2024. doi: 10.1109/ICRA57147.2024.10610848. URL <https://ieeexplore.ieee.org/abstract/document/10610848>. 4.1, 4.2, 4.3.1
 - [69] Haresh Karnan, Siddharth Desai, Josiah P Hanna, Garrett Warnell, and Peter Stone. Reinforced grounded action transformation for sim-to-real transfer. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4397–4402. IEEE, 2020. 3.2.3
 - [70] Pradeep K Khosla and Takeo Kanade. Parameter identification of robot dynamics. In *1985 24th IEEE conference on decision and control*, pages 1754–1760. IEEE,

1985. 3.2.2
- [71] Donghyun Kim, Steven Jens Jorgensen, Jaemin Lee, Junhyeok Ahn, Jianwen Luo, and Luis Sentis. Dynamic locomotion for passive-ankle biped robots and humanoids using whole-body locomotion control. *Int. J. Robotics Res.*, 39(8), 2020. doi: 10.1177/0278364920918014. URL <https://doi.org/10.1177/0278364920918014>. 3.5.3
 - [72] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model, June 2024. URL <http://arxiv.org/abs/2406.09246>. arXiv:2406.09246 [cs]. 4.1
 - [73] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 2.1
 - [74] Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects from sensory data. *The International Journal of Robotics Research*, 41(8):778–797, 2022. 3.2.3
 - [75] F Kozin and HG Natke. System identification techniques. *Structural safety*, 3(3-4): 269–316, 1986. 3.2.2
 - [76] 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*, 40:429–455, 2016. 2.4
 - [77] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021. 3.2.2
 - [78] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid Motor Adaptation for Legged Robots, July 2021. URL <http://arxiv.org/abs/2107.04034>. arXiv:2107.04034 [cs]. 1
 - [79] Ashish Kumar, Zhongyu Li, Jun Zeng, Deepak Pathak, Koushil Sreenath, and Jitendra Malik. Adapting rapid motor adaptation for bipedal robots. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1161–1168. IEEE, 2022. 3.2.2
 - [80] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47): eabc5986, 2020. 3.2.2, 3.2.3

- [81] Kuang-Huei Lee, Ofir Nachum, Tingnan Zhang, Sergio Guadarrama, Jie Tan, and Wenhao Yu. Pi-ars: Accelerating evolution-learned visual-locomotion with predictive information representations. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1447–1454. IEEE, 2022. 3.2.2
- [82] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul McVay, Michael Rabbat, and Yuandong Tian. Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping. August 2024. URL <https://openreview.net/forum?id=SGoVIC0u0f#discussion>. 4.1
- [83] Jinhan Li, Yifeng Zhu, Yuqi Xie, Zhenyu Jiang, Mingyo Seo, Georgios Pavlakos, and Yuke Zhu. Okami: Teaching humanoid robots manipulation skills through single video imitation. In *8th Annual Conference on Robot Learning*, 2024. 1, 2.1
- [84] Tianyu Li, Hartmut Geyer, Christopher G Atkeson, and Akshara Rai. Using deep reinforcement learning to learn high-level policies on the atias biped. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 263–269. IEEE, 2019. 3.2.1
- [85] 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. IEEE, 2021. 3.2.1
- [86] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Robust and versatile bipedal jumping control through reinforcement learning. *Robotics: Science and Systems (RSS)*, 2023. 3.1, 3.2.1
- [87] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *arXiv preprint arXiv:2401.16889*, 2024. 2.1, 2.4, 3.1, 3.2.1, 3.2.2
- [88] Qiayuan Liao, Bike Zhang, Xuanyu Huang, Xiaoyu Huang, Zhongyu Li, and Koushil Sreenath. Berkeley humanoid: A research platform for learning-based control. *arXiv preprint arXiv:2407.21781*, 2024. 2.1, 3.2.1
- [89] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. *arXiv preprint arXiv:2404.16823*, 2024. 1, 2.1
- [90] Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids, 2025. URL <https://arxiv.org/abs/2502.20396>. 1
- [91] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning. November 2023. URL <https://openreview.net/forum?id=w0H2xGH1kw>. 4.1

- [92] Yang Liu, Weixing Chen, Yongjie Bai, Xiaodan Liang, Guanbin Li, Wen Gao, and Liang Lin. Aligning cyber space with physical world: A comprehensive survey on embodied ai, 2024. URL <https://arxiv.org/abs/2407.06886>. 1
- [93] Lennart Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer. 3.1
- [94] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model. *arXiv preprint arXiv:2411.14386*, 2024. 3.1, 3.2.1
- [95] Junfeng Long, Zirui Wang, Quanyi Li, Liu Cao, Jiawei Gao, and Jiangmiao Pang. Hybrid internal model: Learning agile legged locomotion with simulated robot response. In *The Twelfth International Conference on Learning Representations*, 2024. 3.2.2
- [96] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866. 2023. 3.3.1
- [97] Chenhao Lu, Xuxin Cheng, Jialong Li, Shiqi Yang, Mazeyu Ji, Chengjing Yuan, Ge Yang, Sha Yi, and Xiaolong Wang. Mobile-television: Predictive motion priors for humanoid whole-body control. *arXiv preprint arXiv:2412.07773*, 2024. 3.2.1
- [98] Shixin Luo, Songbo Li, Ruiqi Yu, Zhicheng Wang, Jun Wu, and Qiuguo Zhu. Pie: Parkour with implicit-explicit learning framework for legged robots. *IEEE Robotics and Automation Letters*, 2024. 3.2.2
- [99] Zhengyi Luo, Jinkun Cao, Kris Kitani, Weipeng Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023. 3.2.1
- [100] Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. *arXiv preprint arXiv:2310.04582*, 2023. 2.4
- [101] Zhengyi Luo, Jinkun Cao, Alexander W. Winkler, Kris Kitani, and Weipeng Xu. Perpetual humanoid control for real-time simulated avatars. In *International Conference on Computer Vision (ICCV)*, 2023. 2.4
- [102] Zhengyi Luo, Jiashun Wang, Kangni Liu, Haotian Zhang, Chen Tessler, Jingbo Wang, Ye Yuan, Jinkun Cao, Zihui Lin, Fengyi Wang, et al. Smplolympics: Sports environments for physically simulated humanoids. *arXiv preprint arXiv:2407.00187*, 2024. 3.2.1
- [103] Shangke Lyu, Xin Lang, Han Zhao, Hongyin Zhang, Pengxiang Ding, and Donglin Wang. RL2ac: Reinforcement learning-based rapid online adaptive control for legged robot robust locomotion. In *Proceedings of the Robotics: Science and Systems*, 2024.

3.2.2

- [104] Steve Macenski and Ivona Jambrecic. SLAM Toolbox: SLAM for the dynamic world. *Journal of Open Source Software*, 6(61):2783, May 2021. ISSN 2475-9066. doi: 10.21105/joss.02783. URL <https://joss.theoj.org/papers/10.21105/joss.02783>. (document), 4.2, 4.4.2, 4.6
- [105] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The Marathon 2: A Navigation System. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725, October 2020. doi: 10.1109/IROS45743.2020.9341207. URL <http://arxiv.org/abs/2003.00368>. arXiv:2003.00368 [cs]. (document), 4.2, 4.4.2, 4.6
- [106] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019. 2.1, 2.2.3
- [107] 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. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 1, 2.3, 3.2.1, 3.3.1, 3.5
- [108] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *arXiv preprint arXiv:2205.02824*, 2022. 3.1
- [109] Gabriel B Margolis, Xiang Fu, Yandong Ji, and Pulkit Agrawal. Learning to see physical properties with active sensing motor policies. *arXiv preprint arXiv:2311.01405*, 2023. 3.2.2
- [110] Gabriel B Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. *The International Journal of Robotics Research*, 43(4):572–587, 2024. 2.2.2, 3.2.2
- [111] Robert McCarthy, Daniel C. H. Tan, Dominik Schmidt, Fernando Acero, Nathan Herr, Yilun Du, Thomas G. Thuruthel, and Zhibin Li. Towards generalist robot learning from internet video: A survey, 2024. URL <https://arxiv.org/abs/2404.19664>. 1
- [112] 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*, 7(62):eabk2822, 2022. 3.2.2
- [113] Mayank Mittal, Calvin Yu, Qinxu Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics*

- and *Automation Letters*, 8(6):3740–3747, 2023. doi: 10.1109/LRA.2023.3270034. [1](#), [3.2.1](#), [3.5](#)
- [114] Thomas Moore and Daniel Stouch. A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In Emanuele Menegatti, Nathan Michael, Karsten Berns, and Hiroaki Yamaguchi, editors, *Intelligent Autonomous Systems 13*, pages 335–348, Cham, 2016. Springer International Publishing. ISBN 978-3-319-08338-4. doi: 10.1007/978-3-319-08338-4_25. [4.4.2](#)
 - [115] Fabio Muratore, Christian Eilers, Michael Gienger, and Jan Peters. Data-efficient domain randomization with bayesian optimization. *IEEE Robotics and Automation Letters*, 6(2):911–918, 2021. [3.2.2](#)
 - [116] I Made Aswin Nahrendra, Byeongho Yu, and Hyun Myung. Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5078–5084. IEEE, 2023. [3.2.2](#)
 - [117] Kei Okada, Takashi Ogura, Atsushi Haneda, Junya Fujimoto, Fabien Gravot, and Masayuki Inaba. Humanoid motion generation system on hrp2-jsk for daily life environment. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 4, pages 1772–1777. IEEE, 2005. [2.4](#)
 - [118] Michael O’Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, 2022. [3.2.2](#), [3.2.3](#)
 - [119] Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Hao-Shu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao,

- Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booyer, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minh Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Panag R Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundareshan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Bajjal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham Sonawani, Shuran Song, Sichun Xu, Siddhant Halder, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Xu Liangwei, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, and Zipeng Lin. Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903, May 2024. doi: 10.1109/ICRA57147.2024.10611477. URL <https://ieeexplore.ieee.org/document/10611477>. 4.6.1
- [120] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al.

- Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. 2.1
- [121] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL <http://doi.acm.org/10.1145/3197517.3201311>. 2.4
 - [122] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018. 3.1, 3.2.1, 3.3.2, 3.3.2
 - [123] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018. 3.1
 - [124] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions On Graphics (TOG)*, 37(6):1–14, 2018. 3.2.1
 - [125] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020. 3.2.2
 - [126] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022. 2.4
 - [127] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)*, 41(4):1–17, 2022. 3.2.1
 - [128] Haozhi Qi, Ashish Kumar, Roberto Calandra, Yi Ma, and Jitendra Malik. In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning*, pages 1722–1732. PMLR, 2023. 3.2.2
 - [129] 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*, 32(3):397–419, 2015. 1, 2.4
 - [130] I Radosavovic, B Zhang, B Shi, J Rajasegaran, S Kamat, T Darrell, K Sreenath, and J Malik. Humanoid locomotion as next token prediction. arxiv. 2024. *arXiv preprint arXiv:2402.19469*, 2024. 3.1, 3.2.1
 - [131] Ilija Radosavovic, Tete Xiao, Bike Zhang, Trevor Darrell, Jitendra Malik, and

- Koushil Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024. 1, 2.1, 3.1, 3.2.1
- [132] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid Locomotion as Next Token Prediction, February 2024. URL <http://arxiv.org/abs/2402.19469>. arXiv:2402.19469 [cs]. 4.1
- [133] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019. 3.2.2
- [134] Adrian Remonda, Nicklas Hansen, Ayoub Raji, Nicola Musiu, Marko Bertogna, Eduardo Veas, and Xiaolong Wang. A Simulation Benchmark for Autonomous Racing with Large-Scale Human Data, July 2024. URL <http://arxiv.org/abs/2407.16680>. arXiv:2407.16680 [cs]. 4.3.1
- [135] Allen Z Ren, Hongkai Dai, Benjamin Burchfiel, and Anirudha Majumdar. Adapt-sim: Task-driven simulation adaptation for sim-to-real transfer. *arXiv preprint arXiv:2302.04903*, 2023. 3.2.2
- [136] Unitree Robotics. Unitree g1 humanoid agent ai avatar, 2024. URL <https://www.unitree.com/g1>. 3.1, 3.5.3
- [137] Unitree Robotics. Unitree h1 the world’s first full-size motor drive humanoid robot flips on ground, 2024. URL <https://www.youtube.com/watch?v=V1LyWsiTgms>. 3.2.1
- [138] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 2.2.5
- [139] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022. 3.1
- [140] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent asimo: System overview and integration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 3, pages 2478–2483. IEEE, 2002. 2.4
- [141] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza, and Markus Ryll. Real-Time Neural MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms. *IEEE Robotics and Automation Letters*, 8(4):2397–2404, April 2023. ISSN 2377-3766. doi: 10.1109/LRA.2023.3246839. URL <https://ieeexplore.ieee.org/document/10049101>. Conference Name: IEEE Robotics and Automation Letters. 4.4.1

- [142] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. [2.2.1](#), [3.2.1](#), [3.3.2](#)
- [143] Luis Sentis and Oussama Khatib. A whole-body control framework for humanoids operating in human environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2641–2648. IEEE, 2006. [2.4](#)
- [144] Agon Serifi, Ruben Grandia, Espen Knoop, Markus Gross, and Moritz Bächer. Vmp: Versatile motion priors for robustly tracking motion on physical characters. 2024. [2.4](#)
- [145] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. ViNT: A Foundation Model for Visual Navigation, June 2023. URL <https://arxiv.org/abs/2306.14846v2>. [4.1](#)
- [146] Guanya Shi, Xichen Shi, Michael O’Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural lander: Stable drone landing control using learned dynamics. In *2019 international conference on robotics and automation (icra)*, pages 9784–9790. IEEE, 2019. [3.1](#)
- [147] Guanya Shi, Wolfgang Hönig, Xichen Shi, Yisong Yue, and Soon-Jo Chung. Neural-swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, 2021. [3.2.3](#)
- [148] Aditya Shirwatkar, Naman Saxena, Kishore Chandra, and Shishir Kolathaya. Pip-loco: A proprioceptive infinite horizon planning framework for quadrupedal robot locomotion. *arXiv preprint arXiv:2409.09441*, 2024. [3.2.2](#)
- [149] Aayam Shrestha, Pan Liu, German Ros, Kai Yuan, and Alan Fern. Generating physically realistic and directable human motions from multi-modal inputs. In *European Conference on Computer Vision*, pages 1–17. Springer, 2024. [2.4](#)
- [150] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018. [3.2.3](#)
- [151] Matthew Sivaprakasam, Parv Maheshwari, Mateo Guaman Castro, Samuel Triest, Micah Nye, Steve Willits, Andrew Saba, Wenshan Wang, and Sebastian Scherer. TartanDrive 2.0: More Modalities and Better Infrastructure to Further Self-Supervised Learning Research in Off-Road Driving Tasks, February 2024. URL <https://arxiv.org/abs/2402.01913v1>. [4.1](#)
- [152] Kyle Stachowicz and Sergey Levine. RACER: Epistemic Risk-Sensitive RL Enables Fast Driving with Fewer Crashes, May 2024. URL <http://arxiv.org/abs/2405.04714>. arXiv:2405.04714 [cs]. [4.1](#)
- [153] {Stereolabs}. stereolabs/zed-ros2-wrapper, September 2024. URL <https://>

- github.com/stereolabs/zed-ros2-wrapper. original-date: 2019-04-12T14:23:46Z. (document), 4.2, 4.6.2
- [154] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018. 3.1, 3.2.2
 - [155] Chen Tessler, Yoni Kasten, Yunrong Guo, Shie Mannor, Gal Chechik, and Xue Bin Peng. Calm: Conditional adversarial latent models for directable virtual characters. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701597. doi: 10.1145/3588432.3591541. URL <https://doi.org/10.1145/3588432.3591541>. 2.4
 - [156] Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. Masked-mimic: Unified physics-based character control through masked motion. In *ACM Transactions On Graphics (TOG)*. ACM New York, NY, USA, 2024. 2.4
 - [157] Chen Tessler, Yunrong Guo, Ofir Nabati, Gal Chechik, and Xue Bin Peng. Masked-mimic: Unified physics-based character control through masked motion inpainting. *ACM Transactions on Graphics (TOG)*, 43(6):1–21, 2024. 3.2.1, 3.3.1
 - [158] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017. 3.1
 - [159] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109. 3.2.1
 - [160] Unitree. Unitree’s first universal humanoid robot, 2023. URL <https://www.unitree.com/h1>. 2.3, 2.3
 - [161] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html. 4.1
 - [162] Hongxi Wang, Haoxiang Luo, Wei Zhang, and Hua Chen. Cts: Concurrent teacher-student reinforcement learning for legged locomotion. *IEEE Robotics and Automation Letters*, 2024. 3.2.2
 - [163] Huayi Wang, Zirui Wang, Junli Ren, Qingwei Ben, Tao Huang, Weinan Zhang, and Jiangmiao Pang. Beamdojo: Learning agile humanoid locomotion on sparse footholds, 2025. URL <https://arxiv.org/abs/2502.10363>. 1

- [164] Jiashun Wang, Jessica Hodgins, and Jungdam Won. Strategy and skill learning for physics-based table tennis animation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. [3.2.1](#)
- [165] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution, 2024. URL <https://arxiv.org/abs/2409.12191>. [1](#)
- [166] Sean J. Wang, Honghao Zhu, and Aaron M. Johnson. Pay Attention to How You Drive: Safe and Adaptive Model-Based Reinforcement Learning for Off-Road Driving, October 2023. URL <http://arxiv.org/abs/2310.08674>. arXiv:2310.08674 [cs]. [4.1](#), [4.2](#)
- [167] Yinhuai Wang, Jing Lin, Ailing Zeng, Zhengyi Luo, Jian Zhang, and Lei Zhang. Physhoi: Physics-based imitation of dynamic human-object interaction. *arXiv preprint arXiv:2312.04393*, 2023. [3.2.1](#)
- [168] Yinhuai Wang, Qihan Zhao, Runyi Yu, Ailing Zeng, Jing Lin, Zhengyi Luo, Hok Wai Tsui, Jiwen Yu, Xiu Li, Qifeng Chen, et al. Skillmimic: Learning reusable basketball skills from demonstrations. *arXiv preprint arXiv:2408.15270*, 2024. [3.2.1](#)
- [169] Yufu Wang, Ziyun Wang, Lingjie Liu, and Kostas Daniilidis. Tram: Global trajectory and motion of 3d humans from in-the-wild videos. In *European Conference on Computer Vision*, pages 467–487. Springer, 2025. ([document](#)), [3.1](#), [3.3.1](#)
- [170] Zhicheng Wang, Wandi Wei, Ruiqi Yu, Jun Wu, and Qiuguo Zhu. Toward understanding key estimation in learning robust humanoid locomotion. *arXiv preprint arXiv:2403.05868*, 2024. [3.2.2](#)
- [171] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440, May 2016. doi: 10.1109/ICRA.2016.7487277. URL https://ieeexplore.ieee.org/abstract/document/7487277?casa_token=8gyB7VO8MKcAAAAA:Nx9HyqUqpCdvCKEY2xpr8U8Wp5Gufuxvfb0SCJRfw0vY9sIIhQ8zajNWtMUKfOE6_fyCjA_XSQ. [4.1](#), [4.2](#)
- [172] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information theoretic MPC for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, May 2017. doi: 10.1109/ICRA.2017.7989202. URL <https://ieeexplore.ieee.org/document/7989202/?arnumber=7989202>. [4.1](#)
- [173] Lior Wolf. Ai for humanoid robotics - a lecture by mentee robotics’ ceo, prof. lior

- wolf. YouTube video, 2025. URL <https://www.youtube.com/watch?v=y1LG4YwUtoo>. Accessed: 2025-01-31. 3.1
- [174] Jungdam Won, Deepak Gopinath, and Jessica Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39, 2020. ISSN 0730-0301,1557-7368. 2.4
 - [175] Murphy Wonsick and Taşkın Padır. Human-humanoid robot interaction through virtual reality interfaces. In *2021 IEEE Aerospace Conference (50100)*, pages 1–7. IEEE, 2021. 2.4
 - [176] Peilin Wu, Weiwei Xie, Jiahang Cao, Hang Lai, and Weinan Zhang. Loopsr: Looping sim-and-real for lifelong policy adaptation of legged robots. *arXiv preprint arXiv:2409.17992*, 2024. 3.2.2
 - [177] Wenli Xiao, Tairan He, John Dolan, and Guanya Shi. Safe Deep Policy Adaptation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 17286–17292, May 2024. doi: 10.1109/ICRA57147.2024.10611340. URL <https://ieeexplore.ieee.org/abstract/document/10611340>. 4.1, 4.2
 - [178] Wenli Xiao, Haoru Xue, Tony Tao, Dvij Kalaria, John M Dolan, and Guanya Shi. Anycar to anywhere: Learning universal dynamics model for agile and adaptive mobility. *arXiv preprint arXiv:2409.15783*, 2024. 3.1
 - [179] Zhaoming Xie, Patrick Clary, Jeremy Dao, Pedro Morais, Jonathan Hurst, and Michiel Panne. Learning locomotion skills for cassie: Iterative design and sim-to-real. In *Conference on Robot Learning*, pages 317–329. PMLR, 2020. 3.2.1
 - [180] Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. *arXiv preprint arXiv:2409.15610*, 2024. 2.1
 - [181] Haoru Xue, Edward L. Zhu, John M. Dolan, and Francesco Borrelli. Learning Model Predictive Control with Error Dynamics Regression for Autonomous Racing. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13250–13256, May 2024. doi: 10.1109/ICRA57147.2024.10611628. URL <https://ieeexplore.ieee.org/abstract/document/10611628>. 4.1, 4.2, 4.6.1
 - [182] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data, 2024. URL <https://arxiv.org/abs/2401.10891>. 1
 - [183] Yuxiang Yang, Guanya Shi, Changyi Lin, Xiangyun Meng, Rosario Scalise, Mateo Guaman Castro, Wenhao Yu, Tingnan Zhang, Ding Zhao, Jie Tan, et al. Agile continuous jumping in discontinuous terrains. *arXiv preprint arXiv:2409.10923*, 2024. 3.2.2

- [184] Zeji Yi, Chaoyi Pan, Guanqi He, Guannan Qu, and Guanya Shi. CoVO-MPC: Theoretical Analysis of Sampling-based MPC and Optimal Covariance Design, January 2024. URL <http://arxiv.org/abs/2401.07369>. arXiv:2401.07369 [cs]. 4.4.1, 4.4.1
- [185] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017. 3.2.2
- [186] Wenhao Yu, C Karen Liu, and Greg Turk. Policy transfer with strategy optimization. *arXiv preprint arXiv:1810.05751*, 2018. 3.2.2
- [187] Wenhao Yu, Visak CV Kumar, Greg Turk, and C Karen Liu. Sim-to-real transfer for biped locomotion. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3503–3510. IEEE, 2019. 3.1, 3.2.2
- [188] Wenhao Yu, Jie Tan, Yunfei Bai, Erwin Coumans, and Sehoon Ha. Learning fast adaptation with meta strategy optimization. *IEEE Robotics and Automation Letters*, 5(2):2950–2957, 2020. 3.2.2
- [189] YE YUAN and Viktor Makoviychuk. Learning physically simulated tennis skills from broadcast videos. 2023. 3.2.1
- [190] Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground, 2025. URL <https://arxiv.org/abs/2502.08844>. 1
- [191] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. WoCoCo: Learning Whole-Body Humanoid Control with Sequential Contacts. 4.1
- [192] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024. 1, 2, 2.1, 3.2.1
- [193] Qiang Zhang, Peter Cui, David Yan, Jingkai Sun, Yiqun Duan, Gang Han, Wen Zhao, Weining Zhang, Yijie Guo, Arthur Zhang, et al. Whole-body humanoid robot locomotion with human reference. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11225–11231. IEEE, 2024. 3.2.1
- [194] Tony Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023. ISBN 978-0-9923747-9-2. doi: 10.15607/RSS.2023.XIX.016. URL <http://www.roboticsproceedings.org/rss19/p016.pdf>. 4.1
- [195] Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning, 2025. URL <https://arxiv.org/abs/2502.08844>.

[//arxiv.org/abs/2411.04983](https://arxiv.org/abs/2411.04983). 1

- [196] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. *arXiv preprint arXiv:2406.10759*, 2024. 2.1, 2.4, 3.1, 3.2.1
- [197] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Chormanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of The 7th Conference on Robot Learning*, pages 2165–2183. PMLR, December 2023. URL <https://proceedings.mlr.press/v229/zitkovich23a.html>. ISSN: 2640-3498. 4.1