Learning Universal Humanoid Control

Zhengyi Luo

CMU-RI-TR-25-35

April 25, 2025

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Kris Kitani, Chair
Guanya Shi
Shubham Tulsiani
Xue Bin Peng, *Simon Fraser University, NVIDIA*
Karen Liu, *Stanford University*

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

Abstract

Since infancy, humans acquire motor skills, behavioral priors, and objectives by learning from their caregivers. Similarly, as we create humanoids in our own image, we aspire for them to learn from us and develop universal physical and cognitive capabilities that are comparable to, or even surpass, our own. In this thesis, we explore how to equip humanoids with the mobility, dexterity, and environmental awareness necessary to perform meaningful tasks. Unlike previous efforts that focus on learning a narrow set of tasks, such as traversing terrains, imitating a few human motion clips, or playing a single game, we emphasize scaling humanoid control tasks by leveraging large-scale human data (*e.g.*, motion, videos). We show that scaling brings numerous benefits, gradually moving us closer to achieving truly "universal" capabilities.

Our key idea centers around scaling up humanoid motion tracking and forming a foundational humanoid control prior that can be used to speed up task learning. Just like young animals are born with the instinct to walk, run, and grasp, we wish to equip humanoids with motor control priors that lead to human-like movement. We begin by scaling the reinforcement learning based motion tracking framework, enabling humanoids to imitate large-scale kinematic human motion datasets. This motion imitator forms the basis for acquiring motor skills: given a kinematic reference motion, the imitator can robustly control the humanoid to execute everyday activities and more complex, dynamic movements. Such a motion imitator can be used for human pose estimation, teleoperation, and controlling simulated avatars using first-person and third-person cameras.

Equipped with such a motion tracker, we distill behaviors from the tracker into a compact, physics-based control latent space and form a general-purpose humanoid control prior. This prior enables the reuse of previously learned motor skills from a large-scale dataset. Randomly sampling from this latent space leads to human-like behaviors from the humanoid. Leveraging this latent representation in hierarchical reinforcement learning significantly improves sample efficiency and produces human-like motion. A critical aspect of this framework is ensuring that the latent space faithfully encapsulates the full range of motor skills present in the source dataset—a property we verify empirically.

Building upon such a humanoid control prior, we study simulated humanoids equipped with dexterous hands, touch sensors, and vision to interact with their environments and manipulate objects. We find that our control prior significantly simplifies the training process for manipulation tasks, and we can learn policies that generalize across diverse sets of objects and scenes. Along the way, we solve practical problems involved in humanoid dexterous manipulation and perception-int-the-loop control.

Finally, we take a step toward real-world deployment by transferring this framework to physical humanoids. As a first milestone, we train a universal humanoid motion tracker that runs in real time and can be used for humanoid teleoperation. This real-world deployment highlights the practicality of our approach and sets the stage for future work in learning universal controllers for real humanoids.

Dedicated to my family and friends.

Acknowledgments

Pursuing a Ph.D. has been a journey filled with challenges, growth, and countless memorable moments. I am deeply grateful to the advisors, mentors, collaborators, friends, and family who supported me throughout this adventure.

First, I would like to sincerely thank my advisor, Kris Kitani. During my master's studies, Kris's many lessons and dedications helped me grow tremendously as an independent researcher. Then, Kris offered me a Ph.D. position at his lab, where he continued to support and spur me to grow. His honest feedback, critical thinking, and high standards help me step out of my comfort zone. His passion, wisdom, and encouragement have been instrumental throughout my academic journey. I could not think of a better advisor than Kris.

I also want to thank my thesis committee members: Guanya Shi, Shubham Tulsiani, Xue Bin Peng, and Karen Liu, who are pioneers in their respective fields. Their work has laid the foundation for the work presented in this thesis, and it is an honor to have them on my committee.

I am also thankful to my undergraduate advisor, Kostas Daniilidis, who introduced me to the world of research and 3D computer vision. My undergraduate mentors, Georgios Pavlakos and Carlos Esteves, were not only brilliant role models but also incredibly patient and supportive during my early steps into academia.

I have the fortune of visiting Meta for two years and could not thank my mentor and collaborators, Weipeng Xu, Alexander Winkler, Sammy Christen, Jing Huang, Josh Merel, and Yaser Sheikh, enough. I also spent two wonderful internships at NVIDIA Research, mentored by and worked with Sanja Fidler, Xue Bin Peng, Davis Remepe, Haotian Zhang, Viktor Makoviychuk, Or Litany, Chen Tessler, Yunrong Guo, Gal Chechik, Toru Lin, Linxi Fan, and Yuke Zhu.

My labmates and collaborators at CMU have been my support system. Special thanks to Ye Yuan, who took me under his wings and taught me ways of research; to Jinkun Cao, my closest collaborator at KLab, working together on almost every project; to Jinhyung Park and Yuda Song, whom I always turn to for understanding the latest and greatest in research; to Tairan He, my partner in every real humanoid project — together, we ventured into the bittersweet, exhilarating world of humanoid robots. To Wenli Xiao, Chong Zhang, Changliu Liu, Jiashun Wang, Jyun-Ting Song, Junyu Nan, Ryosuke Hori, Jessica Hodgins, Shun Iwase, Yu-Jhe Li, Rawal Khirodkar, Alireza Golestaneh, Erica Weng, Navyata Sanghvi, Xiaofang Wang, Xinshuo Weng, Vivek Roy, Ryo Hachiuma, Shin Usami, Scott Sun, Kangni Liu, Yunze Man, Shengcao Cao and many others — thank you for your friendship, kindness, and warmth.

Many outside collaborators and friends along the way — Jingbo Wang, Guy Tevet, Michiel van de Panne, Zhiyang Dou, Yan Wu, Siyu Tang, Ailing Zeng, and Yinhuai Wang — thank you for the insightful discussions, inspiring collaborations, and shared enthusiasm that made research more exciting and meaningful.

I am deeply thankful for my partner, Zihui Lin, whose love brought endless light and laughter into my life when I needed it most. I am endlessly grateful to have her by my side.

Finally, I would like to thank my parents, Shu Gao and Xiaolin Luo. Their unwavering support and patience have formed the backbone of my identity. Ten years of studying abroad would not have been possible without their endless encouragement, understanding, and love. Every step I have taken on this journey has been built upon the foundation they provided.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

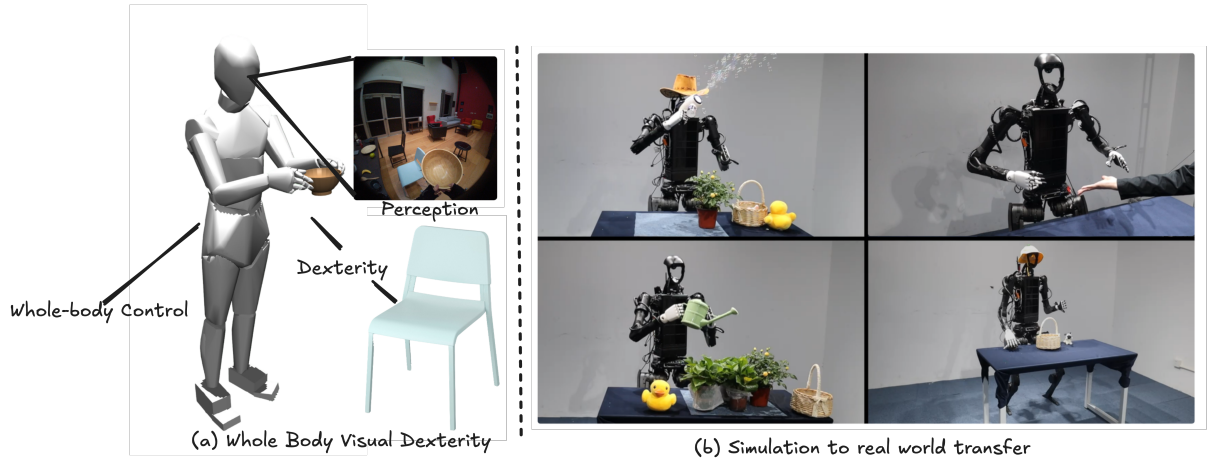


Figure 1.1. We aim to scale up humanoid capabilities in both simulation and the real world, focusing on whole-body control, perception, and dexterity.

For a long time, humanoids have captivated the public's imagination as one of the most promising physical embodiments of general intelligence, largely due to their resemblance to the human form. Simulated humanoids capable of producing human-like motion can be used in graphics and animation, reducing the need for artists to author animations or rely on Motion Capture (MoCap). They can also offer insights into human movement and serve as a general-purpose platform for studying virtual intelligence. Real humanoids, on the other hand, have been envisioned as part of the labor force, tackling tasks ranging from daily household chores to hazardous, high-risk scenarios like search and rescue. In this thesis, we enhance the capabilities of humanoids from the ground up by addressing three core aspects of humanoid control: (1) Mobility, which encompasses body movement excluding the hands; (2) Dexterity, referring to the use of dexterous hands for manipulating objects and interacting with the environment; and (3) Perception, which involves controlling humanoids based on sensory inputs. While prior research in humanoid control has focused on isolated tasks [15, 208, 210, 213, 215, 224, 275] and curated datasets [292, 296, 325], we focus on scaling up learning to obtain "universal" capabilities.

One of our key insights is the emphasis on forming a foundational motor skill latent space via

the task of kinematic motion tracking for humanoids. Although prior efforts have employed reinforcement learning (RL) to imitate single motion clips [208] or thousands of clips [296], these works primarily focused on generating visually realistic movement for graphics and animation. In contrast, we position motion imitation as a tool for acquiring generalizable motor skills. By scaling the motion imitation task to large-scale human motion datasets, such as AMASS [176], we aim to train a single policy capable of imitating all motion sequences, allowing humanoids to achieve human-level agility. To ensure performance, we introduce explicit metrics like success rate and joint position accuracy to evaluate the imitation capability. Our framework marks *the first time* a tracking policy has been able to track an entire large-scale dataset with high precision.

Using our universal motion tracking framework, we distill the tracker into a universally applicable latent motion representation, which serves as a prior for downstream tasks requiring perception (e.g., humanoid terrain traversal, human-object interactions). For a good motor control prior, we identify three key factors 1) random samples from the latent space should lead to human-like motion; 2) the motor latent space should cover all of the motor skills required to perform a large-scale motion dataset; 3) the latent space should speed up downstream task learning by providing a hierarchical action space. Following these three main criteria, we use a teacher-student framework to develop a universal humanoid motion representation for physics-based control by distilling the motor skills from a tracker to a latent space. Our experiments verify that our latent space meets the above three criteria.

Equipped with our universal humanoid motion representation, we study meaningful tasks for humanoids such as dexterous manipulation and visual-dexterous whole-body control. We first design a controller that can grasp diverse objects and follow diverse object trajectories (such as calligraphy mid-air) using state-space policies. We find that leveraging the expressive motion latent space designed beforehand drastically simplifies this dexterous manipulation task in simulation, and we are able to train grasping policies using simple rewards that do not require paired human and object manipulation data. Then, we remove the dependency on knowing the object shape and pose, and train a perception-only policy for visual grasping in diverse kitchen scenarios. In this setting, we involve active perception, whole-body control, and dexterous manipulation, and train multi-task policies to search for objects, grasp, move, and place objects, as well as manipulate articulated objects. To support multi-task visual learning, we design a perception-as-interface framework to remove the dependency of task variables and use visual cues as input for policy learning.

Finally, we take the first step of deploying our simulated humanoid framework to real humanoids. To start, we train universal tracking policies on commercially available humanoids for teleoperation and show that our tracking framework can be used for real-humanoid learning. We solve practical challenges such as motion retargeting from human to humanoid, motion selection and verification, domain randomization, and state-space design for sim-to-real transfer.

1.1 Thesis Overview

In this thesis, we present a framework for scaling up the capabilities needed to make humanoids useful (Figure 1.1). We focus on increasing capabilities in three main aspects as shown in Fig-

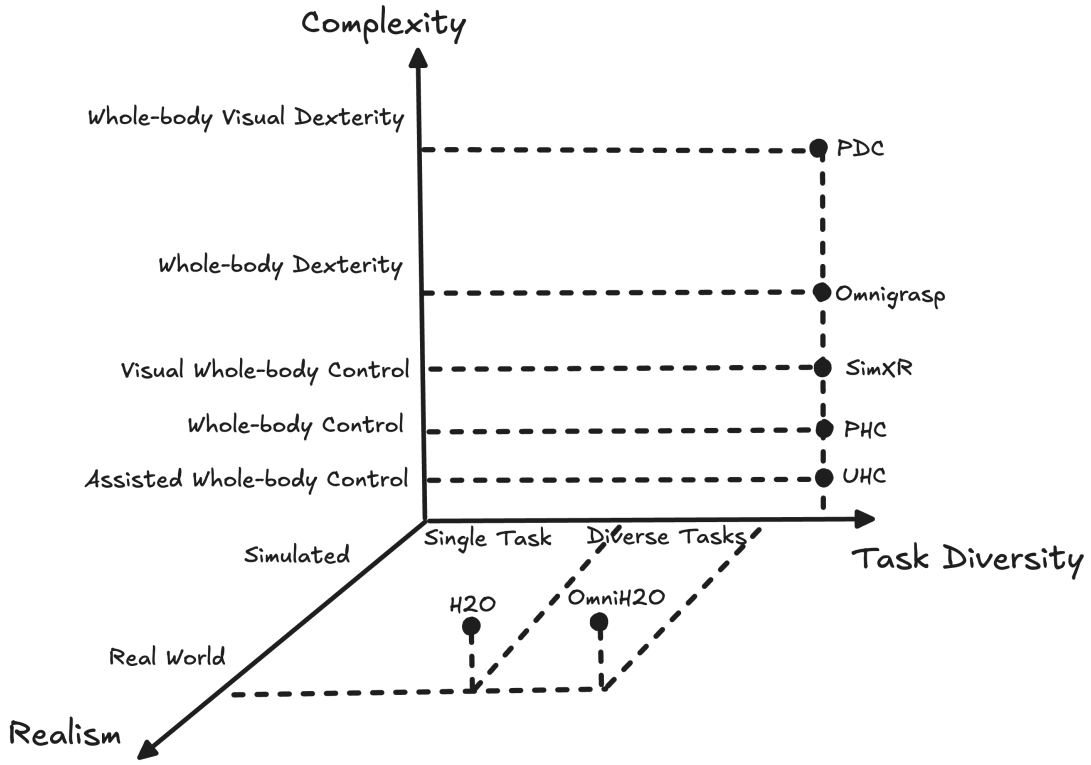


Figure 1.2. The three main aspects of humanoid capabilities that we focus on: complexity, task diversity, and realism.

ure 1.2. The first axis is complexity, where we go from humanoid whole-body control to visual dexterous whole-body control. The second axis is task diversity, going from solving single tasks to diverse tasks. Finally, we aim to tackle both simulated and real-world humanoid tasks. In this chapter, we will first set up the preliminaries of reinforcement learning. Then, in the following chapters, we provide a three-step procedure to scale up these three axes:

- **Part I - Scaling Motion Imitation (Chapter 2-4):** We present scaling up humanoid motion imitation to large-scale human motion datasets. This task serves as the foundation to equip humanoids with the ability to move like humans. In this part, we also utilize this humanoid controller to control simulated avatars from first-person videos.
- **Part II - Forming Foundational Motion Priors (Chapter 5):** We explore learning a universal humanoid motion representation that can be used as a foundational behavior prior for training downstream RL tasks. Equipped with this prior, humanoids can explore like humans when training using RL.
- **Part III - Learning Dexterous and Perceptive Controllers (Chapter 6-7) :** In this part, we leverage our behavioral foundation model to tackle dexterous and perceptive humanoid locomanipulation tasks.
- **Part IV - Transferring to Real Humanoids (Chapter 8-9):** In this part, we take the first step towards transferring to real-world humanoids. Specifically, we tackle scaling up humanoid motion imitation for real humanoids.

Chapter 2. We present Universal Humanoid Controller (UHC), our first attempt at building a humanoid controller that can imitate a large-scale human motion dataset. Due to the diversity of human motion and the challenge of building a single policy that can track all of the motion from a large-scale dataset, we utilize Residual Force Control (RFC) and apply a helping force in the humanoid's root to stabilize training. Though applying this imaginary force compromises the physical realism, it can be helpful in animation and computer vision applications (such as pose estimation). We apply UHC to egocentric pose estimation and estimate both human pose and human-scene interactions. This work is published as: Luo, Zhengyi, Ryo Hachiuma, Ye Yuan, and Kris Kitani. "Dynamics-regulated kinematic policy for egocentric pose estimation." In Proceedings of Advances in Neural Information Processing Systems (NeurIPS 2021) [172].

Chapter 3. Following UHC, we present Perpetual Humanoid Controller (PHC), a humanoid motion imitator capable of imitating a large corpus of motion with high fidelity *without the use of residual force*. To scale to large datasets, we propose a progressive multiplicative compositional policy (PMCP) to combat the forgetting problem when training on a large number of motion sequences. We also equip PHC with the ability to recover from failure/fallen state and resume motion imitation. Equipped with this ability, PHC is ideal for simulated avatar use cases where we no longer require reset during unexpected events. We also pair PHC with a real-time pose estimator to show that it can be used in a video-based avatar use case, where the simulated avatar imitates motion performed by the actors perpetually without requiring reset. Additionally. we connect PHC to a language-based motion generator to demonstrate its ability to mimic generated motion from text. PHC serve as the foundational mobility provider for the following chapters. This work is published as: Luo, Zhengyi, Jinkun Cao, Kris Kitani, and Weipeng Xu. "Perpetual humanoid control for real-time simulated avatars." In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2023) [169].

Chapter 4. We also study simulated humanoid control driven by sensor input (images and headset tracking) captured by an XR headset (SimXR). Here, we explore visual whole-body control, where egocentric visual signals instruct the movement of a simulated humanoid. We propose a simple yet effective end-to-end learning framework that learns to map from headset pose and camera input from XR headsets to humanoid control signals through distillation. Training only using synthetic data, our network can control simulated avatars using real-world data capture from VR headsets with high accuracy in real-time. This work is published as: Luo, Zhengyi, Jinkun Cao, Rawal Khirodkar, Alexander Winkler, Kris Kitani, and Weipeng Xu. "Real-Time Simulated Avatar from Head-Mounted Sensors." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, (CVPR 2024) [168].

Chapter 5. Motion imitation methods typically rely on ground-truth kinematic data during training, which is often unavailable in human-scene interaction tasks due to limited data. Additionally, the active sensing and human-object interaction required in such tasks are not supported by standard motion imitators. How can we leverage the motor skills learned by a pre-trained motion imitator for downstream reinforcement learning (RL) tasks in these contexts? We present a Physics-based UniversaL motion repreSEntation (PULSE) that encompasses a comprehensive range of motor skills for physics-based humanoid control. Our motion representation is learned by distilling skills directly from an imitator. This is achieved by using an

encoder-decoder structure with a variational information bottleneck. Additionally, we jointly learn a prior conditioned on proprioception (humanoid's own pose and velocities) to improve model expressiveness and sampling efficiency for downstream tasks. By sampling from the prior, we can generate long, stable, and diverse human motions. Using this latent space for hierarchical RL, we demonstrate that our policies exhibit human-like behavior when solving tasks. We demonstrate the effectiveness of our motion representation by solving generative tasks (*e.g.* strike, terrain traversal) and motion tracking using VR controllers. This work is published as: Luo, Zhengyi, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. "Universal humanoid motion representations for physics-based control." In Proceedings of the International Conference on Learning Representations (ICLR 2024) [170].

Chapter 6. Equipped with our universal humanoid motion representation as the foundational behavior prior, we begin to tackle dexterous whole-body control tasks. In this chapter, we investigate the problem of controlling a simulated humanoid to grasp various objects and follow diverse trajectories (Omnigrasp). We do not yet consider vision-based policy and uses the object state (*e.g.* pose and shape of the object) provided by the simulator Controlling a simulated humanoid to follow diverse object trajectories is challenging due to the lack of paired MoCap whole-body and object data. Adding dexterous hands to the humanoid also significantly increases the degree of freedom ($69 \rightarrow 153$). As such, the dexterous manipulation task has been mostly approached using a disembodied hand, which does not require whole-body coordination. As the object is passive, a simulated humanoid can easily knock it over during exploration, making it a difficult RL exploration problem. We extend our previously learned motion representation to the dexterous humanoid setting and show that it significantly simplifies the RL learning problem. With the reward consisting only of an "approach" and "object-trajectory following" parts, we can train a humanoid to grasp diverse objects and follow trajectories. This work is published as: Luo, Zhengyi, Jinkun Cao, Sammy Christen, Alexander Winkler, Kris Kitani, and Weipeng Xu. "Omnigrasp: Grasping diverse objects with simulated humanoids." In Proceedings of Advances in Neural Information Processing Systems (NeurIPS 2024) [167].

Chapter 7. In this chapter, we take one more step on the complexity axis and tackle Visual Dexterous whole-body Control (PDC). Using a simulated humanoid that is equipped with proprioception and visual sensors, we aim to control the humanoid to complete locomanipuation tasks in the kitchen. A vision-centric policy introduces significant challenges, as the humanoid no longer knows where the object is and requires active perception to search for objects of interest. Another challenge is task specification when trying to solve multiple different tasks. Each task requires its own state variables, leading to a non-scalable approach for task specification. To tackle these issues, we propose perception-as-interface paradigm where we specify all task variables via perception. Replacing task variables, we provide the humanoid with visual cues (such as visual markers) to indicate task stages and phases. Through large-scale visual RL, we are able to train policies to perform pick and place in the kitchen scene as well as opening drawers. Through RL, behaviors such as searching for objects and gazing emerge. This work is currently in submission as Luo, Zhengyi, Chen Tessler, Toru Lin, Ye Yuan, Tairan He, Wenli Xiao, Yunrong Guo, Gal Chechik, Kris Kitani, Linxi "Jim" Fan, Yuke Zhu. "Emergent Active Perception and Dexterity of Simulated Humanoids from Visual Reinforcement Learning".

Chapter 8. We test our motion imitation framework on a real-world full-sized humanoid [282]. We introduced Human to Humanoid (H2O), a scalable learning-based framework that enables real-time whole-body humanoid robot teleoperation using just an RGB camera. We apply PHC's training pipeline and philosophy and apply them to a real humanoid and design a novel "sim-to-data" process to address the challenge of translating human motion into actions a humanoid robot can perform. We show that, using sim-to-real transfer techniques, we can largely transfer PHC's framework to a real humanoid. This work is published as: He, Tairan*, Zhengyi Luo*, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. "Learning human-to-humanoid real-time whole-body teleoperation." In Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2024) [98].

Chapter 9. In this chapter, we build upon H2O and introduces OmniH2O for dexterous and universal humanoid teleoperation. OmniH2O adopts a teacher-student framework where we first train a teacher policy that has access to privileged information not accessible in the real-world (such as global position and linear velocities). Then, we distill the teacher policy into a sim-to-real ready policy (using the same distillation pipeline used in SimXR and PULSE). The resulting policy can be used to teleoperate a real humanoid to perform diverse dexterous whole-body control tasks (hand controlled separately). Using kinematic pose as the universal interface, it can also interface with text-to-motion models and foundation models to control humanoids autonomously.

Chapter 10. In the final chapter, we summarize our learnings throughout our journey of scaling humanoid capabilities and provide a perspective on how to build behavioral foundational models for humanoid agents. We reflect on why humanoids are the ideal platform for embodied intelligence research, outline the key steps for constructing scalable behavioral priors, and discuss future directions to advance robustness, perception, dexterity, and continual learning toward general-purpose humanoid control.

1.2 Excluded Research

To form a coherent narrative and due to space limitations, I excluded a significant portion of projects during my Ph.D. Below are some of the notable ones:

- **Video Pose Estimation:** Estimation smooth human motion from videos, MEVA. I also developed EmbodiedPose [173], which can estimate physics-based 3D human pose and human-scene interactions from monocular videos, leveraging UHC as the motion tracker.

- **Pedestrian Animation Controller:** Similarly, before developing PHC, PACER was developed for pedestrian animation. PACER can control simulated humanoids to follow trajectories and traverse diverse terrains. Its followup, PACER+, adds the ability to control upper body pose to the controller.

- **Humanoid Sports Skills:** Humanoid playing basketball (SkillMimic) [294] and Olympic Sports (SMPLOlympics) [174].

- **Text to Humanoid Control:** CLoSD [276] and UniPhys [300] combine diffusion with PHC and PULSE, respectively, for controlling simulated humanoids with text commands.
- **Real Humanoid Controller:** HOVER [99] develops a unified way to accommodate different control signals for real humanoid control, and ASAP [96] develops an agile humanoid controller for athletic moves.

1.3 Reinforcement Learning for Humanoid Control

Throughout this thesis, we will follow the general framework of goal-conditioned RL. We aim to obtain a goal-conditioned policy π to control humanoids based on task input. The framework is formulated as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$ of states, actions, transition dynamics, reward function, and discount factor. Physics simulation determines the state $\boldsymbol{s}_t \in \mathcal{S}$ and the dynamics of the transition $\mathcal{T}$, where a policy computes the action $\boldsymbol{a}_t$. For humanoid control tasks, the state $\boldsymbol{s}_t$ contains the proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$. Proprioception is defined as $\boldsymbol{s}_t^{\text{p}} \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$, which contains the 3D body pose $\boldsymbol{q}_t$ and the velocity $\dot{\boldsymbol{q}}_t$. The goal state $\boldsymbol{s}_t^{\text{g}}$ is defined based on the task. Based on proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$, the reward $r_t = \boldsymbol{\mathcal{R}}(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ is used to train the policy using RL (*e.g.* PPO [245]). If a reference action $\hat{\boldsymbol{a}}_t$ can be provided (often by a pre-trained expert), we can also optimize the policy π through policy distillation [238, 241, 244].

To represent human pose and humanoid state, we define body pose $\boldsymbol{q}_t \triangleq (\boldsymbol{\theta}_t, \boldsymbol{p}_t)$ as 3D joint rotation $\boldsymbol{\theta}_t \in \mathbb{R}^{J\times 6}$ (using the 6D rotation representation [346]) and position $\boldsymbol{p}_t \in \mathbb{R}^{J\times 3}$ of all J links on the articulated humanoid. Body velocities $\dot{\boldsymbol{q}}_{1:T}$ are expressed as $\dot{\boldsymbol{q}}_t \triangleq (\boldsymbol{\omega}_t, \boldsymbol{v}_t)$, consisting of angular $\boldsymbol{\omega}_t \in \mathbb{R}^{J\times 3}$ and linear velocities $\boldsymbol{v}_t \in \mathbb{R}^{J\times 3}$. As a notation convention, we use $\tilde{\cdot}$ to represent kinematic quantities (without physics simulation) from pose estimator/keypoint detectors, $\hat{\cdot}$ to denote ground truth quantities from Motion Capture (MoCap), and normal symbols without accents for values from the physics simulation. We use "imitate", "track", and "mimic" reference motion interchangeably

1.4 Humanoid Setup

Parametric Human Model and Human Motion Dataset. Popular in the vision and graphics community, parametric human models such as SMPL [160] are easy to work with representations of human shapes and motions. SMPL represents the human body as body shape parameters $\boldsymbol{\beta} \in \mathcal{R}^{10}$, pose parameters $\boldsymbol{\theta} \in \mathcal{R}^{24\times 3}$, and root translation $\boldsymbol{p} \in \mathcal{R}^{24\times 3}$. Given $\boldsymbol{\beta}$, $\boldsymbol{\theta}$ and $\boldsymbol{p}$, $\mathcal{S}$ denotes the SMPL function, where $\mathcal{S}(\boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{p}) : \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{p} \rightarrow \mathcal{R}^{6890\times 3}$ maps the parameters to the position of the vertices of a triangular human mesh of 6890 vertices. The AMASS [176] dataset contains 40 hours of motion capture expressed in the SMPL parameters.

In this thesis, we consider three different types of humanoids, as shown in Figure 1.3. The SMPL [160] humanoid follows the same kinematic structure as the popular SMPL human model, and we can directly use the joint angles from the SMPL humanoid to animate a human mesh model. Each joint on the SMPL humanoid has three degrees of freedom (DoF). To support

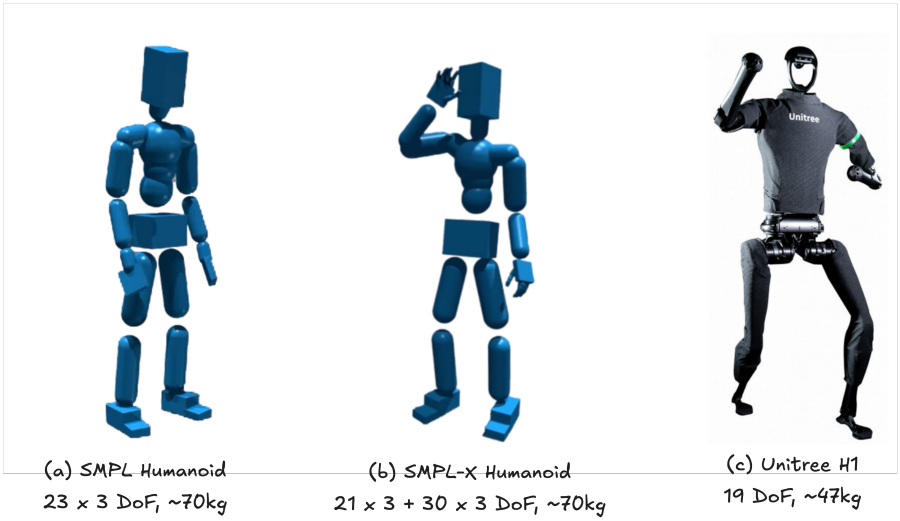


Figure 1.3. We consider three types of humanoid: (a) the SMPL humanoid, which has 23 joints, each of which has 3 DoF; (b) SMPL-X humanoid, which adds fingers to the SMPL humanoid; (c) the Unitree H1 humanoid [282], a real-world humanoid that has 19 DoF.

dexterity, we also use the SMPL-X [204] humanoid, which adds 28 additional joints that belong to the hands. Our motivation for using the SMPL humanoid is its affinity with the visual and graphic community, where the SMPL body model has been widely used for pose estimation [23, 125, 126, 132, 143, 146, 171, 336], motion generation [277, 328, 338], etc. The largest aggregate of kinematic human motion datasets in academia, AMASS [176], is also in the SMPL format.

To control the humanoid, each DoF on the humanoid is controller by a PD controller, and our policy π computes target joint angles for the PD controller. Specifically, the torque τ for the i^{th} DoF is computed according to

$$\tau^i = k^p \circ (\theta_i^d - \theta_i) - k^d \times \dot{\theta}_i \tag{1.1}$$

where k^p and k^d are manually specified gains. θ_i^d is the desired target angle specified by the policy, and θ_i is the current joint angle for the i^{th} DoF.

Part I

Acquiring Motor Skills by Motion Imitation

Chapter 2

UHC: Universal Humanoid Controller for Egocentric Pose Estimation

Figure 2.1. From egocentric videos, we infer physically-plausible 3D human pose and human-object interaction.

2.1 Introduction

Motion imitation forms the foundation for acquiring motor skills throughout this thesis. The task of motion imitation involves translating kinematic motion into dynamic actions, enabling the controlled humanoid to mimic, track, or closely imitate a reference motion. Physics-based motion imitation has gained significant attention in the vision and graphics communities due to its potential to create realistic human motion, facilitate plausible interactions with the environment, and advance future virtual avatar technologies. However, controlling high-degree-of-freedom (DOF) humanoids in simulation presents substantial challenges—they can easily fall, trip, deviate from reference motions, and struggle to recover. Additionally, human motion is incredibly diverse, ranging from simple standing to complex actions like walking, jumping,

backflips, and cartwheels. In this chapter, we take our first step toward scaling up humanoid motion imitation, and in later chapters, we demonstrate that a scaled motion imitator can provide foundational motor skills for a wide range of humanoid tasks.

Our first attempt at scaling up humanoid motion tracking is motivated by the task of physics-based egocentric pose estimation. From a video captured by a single head-mounted wearable camera (*e.g.*, smartglasses, action camera, body camera), we aim to infer the wearer's global 3D full-body pose and interaction with objects in the scene, as illustrated in Fig. 2.1. This is important for applications like virtual and augmented reality, sports analysis, and wearable medical monitoring, where third-person views are often unavailable and proprioception algorithms are needed for understanding the actions of the camera wearer. However, this task is challenging since the wearer's body is often unseen from a first-person view and the body motion needs to be inferred solely based on the videos captured by the *front-facing* camera. Furthermore, egocentric videos usually capture the camera wearer interacting with objects in the scene, which adds additional complexity in recovering a pose sequence that agrees with the scene context. Despite these challenges, we show that it is possible to infer accurate human motion and human-object interaction from a single head-worn front-facing camera.

Egocentric pose estimation can be solved using two different paradigms: (1) a kinematics perspective and (2) a dynamics perspective. *Kinematics-based approaches* study motion without regard to the underlying forces (*e.g.*, gravity, joint torque) and cannot faithfully emulate human-object interaction without modeling proper contact and forces. They can achieve accurate pose estimates by directly outputting joint angles but can also produce results tmahat violate physical constraints (*e.g.* foot skating and ground penetration). *dynamics-based approaches*, or *physics-based approaches*, study motions that result from forces. They map directly from visual input to control signals of a human proxy (humanoid) inside a physics simulator and recover 3D poses through simulation. These approaches have the crucial advantage that they output physically-plausible human motion and human-object interaction (*i.e.*, pushing an object will move it according to the rules of physics). However, since no joint torque is captured in human motion datasets, physics-based humanoid controllers are hard to learn, generalize poorly, and are actively being researched [208, 292, 297, 327].

In this work, we argue that a *hybrid* approach merging the kinematics and dynamics perspectives is needed. Leveraging a large human motion database [176], we learn a task-agnostic dynamics-based humanoid controller to mimic broad human behaviors, ranging from every day motion to dancing and kickboxing. The controller is *general-purpose* and can be viewed as providing low-level motor skills of a human. After the controller is learned, we train an object-aware kinematic policy to specify the target poses for the controller to mimic. One approach is to let the kinematic model produce target motion *only* based on the visual input [292, 325, 329]. This approach uses the physics simulation as a post-processing step: the kinematic model computes the target motion separately from the simulation and may output unreasonable target poses. We propose to synchronize the two aspects by designing a kinematic policy that guides the controller and receives timely feedback through comparing its target pose and the resulting simulation state. Our model thus serves as a high-level motion planning module that adapts intelligently based on the current simulation state. In addition, since our kinematic policy only

outputs poses and does not model joint torque, it can receive direct supervision from motion capture (MoCap) data. While poses from MoCap can provide an initial-guess of target motion, our model can search for better solutions through trial and error. This learning process, dubbed *dynamics-regulated training*, jointly optimizes our model via supervised learning and reinforcement learning, and significantly improves its robustness to real-world use cases.

In summary, our contributions are as follows: (1) we are the first to tackle the challenging task of estimating physically-plausible 3D poses and human-object interactions from a single front-facing camera; (2) we learn a general-purpose humanoid controller from a large MoCap dataset and can perform a broad range of motions inside a physics simulation; (3) we propose a dynamics-regulated training procedure that synergizes kinematics, dynamics, and scene context for egocentric vision; (4) experiments on a controlled motion capture laboratory dataset and a real-world dataset demonstrate that our model outperforms other state-of-the-art methods on pose-based and physics-based metrics, while generalizing to videos taken in real-world scenarios.

2.2 Related Work

Third-person human pose estimation. The task of estimating the 3D human pose (and sometimes shape) from *third-person* video is a popular research area in the vision community [23, 82, 86, 91, 125, 132, 135, 171, 187, 205, 230, 236, 313, 329, 333], with methods aiming to recover 3D joint positions [12, 145, 205, 272], 3D joint angles with respect to a parametric human model [23, 125, 132, 171], and dense body parts [86]. Notice that all these methods are purely kinematic and disregard physical reasoning. They also do not recover the global 3D root position and are evaluated by zeroing out the body center (root-relative). A smaller number of works factor in human [29,232,249,251,284,329] through postprocessing, physics-based trajectory optimization, or using a differentiable physics model. These approaches can produce physically-plausible human motion, but since they do not utilize a physics simulator and does not model contact, they can not faithfully model human-object interaction. SimPoE [329], a recent work on third-person pose estimation using simulated character control, is most related to ours, but 1) trains a single and dataset-specific humanoid controller per dataset; 2) designs the kinematic model to be independent from simulation states.

Egocentric human pose estimation. Compared to third-person human pose estimation, there are only a handful of attempts at estimating 3D full body poses from egocentric videos due to the ill-posed nature of this task. Most existing methods still assume partial visibility of body parts in the image [235,281,310], often through a downward-facing camera. Among works where the human body is mostly not observable [121,195,324,325], Jiang *et al*. [121] use a kinematics-based approach where they construct a motion graph from the training data and recover the pose sequence by solving the optimal pose path. Ng *et al*. [195] focus on modeling person-to-person interactions from egocentric videos and inferring the wearer's pose conditioning on the other person's pose. The works most related to ours are [117, 324, 325] which use dynamics-based approaches and map visual inputs to control signals to perform physically-plausible human motion inside a physics simulation. They show impressive results on a set of noninteractive lo-

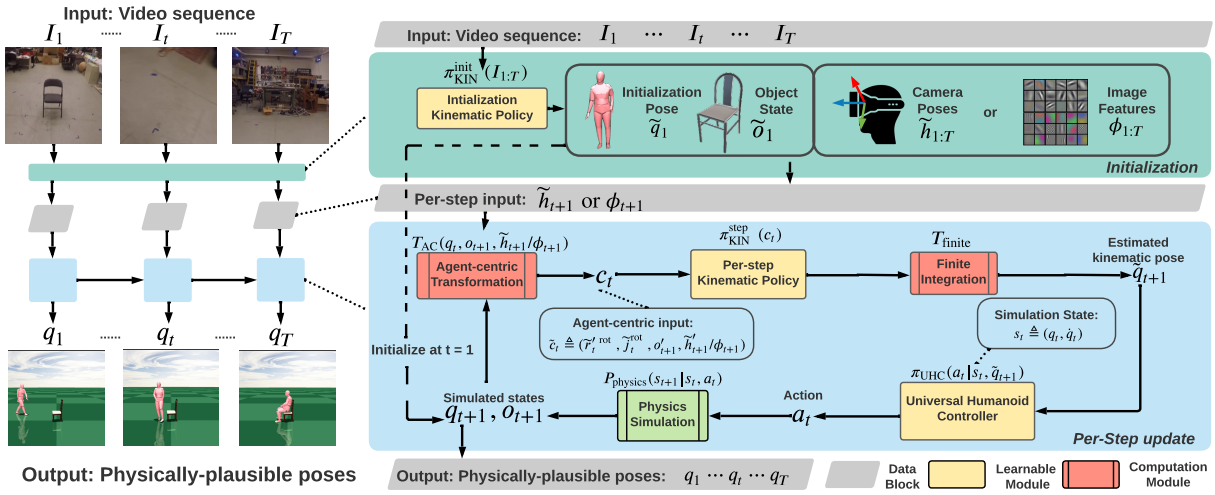


Figure 2.2. Overview of our dynamics-regulated kinematic policy. Given an egocentirc video $\boldsymbol{I}_{1:T}$, our initialization module $\boldsymbol{\pi}_{\text{KIN}}^{\text{init}}(\boldsymbol{I}_{1:T})$ computes the first-frame object state $\widetilde{\boldsymbol{o}}_1$, human pose $\widetilde{\boldsymbol{q}}_1$, camera poses $\widetilde{\boldsymbol{h}}_{1:T}$ or image features $\boldsymbol{\phi}_{1:T}$. The object state $\widetilde{\boldsymbol{o}}_1$ and human pose $\widetilde{\boldsymbol{q}}_1$ are used to initialize the phsycis simulation. At each time step, we roll out our per-step kinematic policy $\boldsymbol{\pi}_{\text{KIN}}^{\text{step}}$ together with the Universal Humanoid Controller to output physically-plausible pose $\boldsymbol{q}_t$ inside a physics simulator.

comotion tasks, but also observe large errors in absolute 3D position tracking–mapping directly from the visual inputs to control signals is a noisy process and prone to error accumulation. In comparison, our work jointly models kinematics and dynamics, and estimates a wider range of human motion and human-object interactions while improving absolute 3D position tracking. To the best of our knowledge, we are the first approach to estimate the 3D human poses from egocentric video while factoring in human-object interactions.

Humanoid control inside physics simulation. Our work is also connected to controlling humanoids to mimic reference motion [38, 46, 46, 208, 214, 297, 327] and interact with objects [38, 181] inside a physics simulator. The core motivation of these works is to learn the necessary dynamics to imitate or generate human motion in a physics simulation. Deep RL has been the predominant approach in this line of work since physics simulators are typically not end-to-end differetiable. Goal-oriented methods [18, 38, 181] does not involve motion imitation and are evaluated on task completion (moving an object, sitting on a chair, moving based on user-input *etc.*). Consequently, these frameworks only need to master a subset of possible motions for task completion. People, on the other hand, have a variety of ways to perform actions, and our agent has to follow the trajectory predefined by egocentric videos. Motion imitation methods [208,214,297,297,327] aim to control characters to mimic a sequence of reference motion, but have been limited to performing a single clip [208, 214, 297, 327] or high-quality MoCap [297] motion (and requires fine-tuning to generalize to other motion generators). In contrast, our dynamics controller is general and can be used to perform everyday motion and human-object interactions estimated by a kinematic motion estimator without task-specific fine-tuning.

2.3 Method

The problem of egocentric pose estimation can be formulated as follows: from a wearable camera footage $\boldsymbol{I}_{1:T}$, we want to recover the wearer's ground truth global 3D poses $\widehat{\boldsymbol{q}}_{1:T}$. Each pose $\widehat{\boldsymbol{q}}_t \triangleq (\widehat{\boldsymbol{r}}_t^{\text{pos}}, \widehat{\boldsymbol{r}}_t^{\text{rot}}, \widehat{\boldsymbol{j}}_t^{\text{rot}})$ consists of the root position $\widehat{\boldsymbol{r}}_t^{\text{pos}}$, root orientation $\widehat{\boldsymbol{r}}_t^{\text{rot}}$, and body joint angles $\widehat{\boldsymbol{j}}_t^{\text{rot}}$ of the human model. Here we adopt the popular SMPL [161] human model and the humanoid we use in physics simulation is created from the kinematic structure and mean body shape defined by SMPL. Our framework first learns a Universal Humanoid Controller (UHC) from a large MoCap dataset (Sec. 2.3.1). The learned UHC can be viewed as providing the lower level muscle skills of a real human, trained by mimicking thousands of human motion sequences. Using the trained UHC, we learn our kinematic policy (Sec. 2.3.2) through dynamics-regulated training (Sec. 2.3.3). At the test time, the kinematic policy provides per-step target motion to the UHC, forming a closed-loop system that operates inside the physics simulation to control a humanoid. The result of the UHC and physics simulation is then used as input to the kinematic policy to produce the next-frame target motion, as depicted in Fig. 2.2. As a notation convention, we use $\widetilde{\cdot}$ to denote kinematic quantities (obtained without using physics simulation), $\widehat{\cdot}$ to denote ground truth quantities, and normal symbols without accents to denote quantities from the physics simulation.

2.3.1 dynamics Model - Universal Humanoid Controller (UHC)

To learn a task-agnostic dynamics model that can be tightly integrated with a kinematic model, we design our controller's state space to only rely on the current simulated pose $\boldsymbol{q}_t$ and target posed $\widehat{\boldsymbol{q}}_{t+1}$ and remove all phase or sequence-level information found in prior arts [208,214,327]. This design allows us to train on an order of magnitude larger dataset of human motion [176] with only pose information and significantly improve our models' ability to mimic diverse and unseen motions. Formally, we model controlling a humanoid to follow a reference motion $\widehat{\boldsymbol{q}}_{1:T}$ as a Markov Decision Process (MDP) defined as a tuple $\mathcal{M} = \langle S, A, P_{\text{physics}}, R, \gamma \rangle$ of states, actions, transition dynamics, reward function, and a discount factor. The state S, reward R, and transition dynamics P_{physics} are provided by the physics simulator, while action A is computed by the policy $\boldsymbol{\pi}_{\text{UHC}}$. At each timestep t, the agent in state $\boldsymbol{s}_t$ takes an action sampled from the policy $\boldsymbol{\pi}_{\text{UHC}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \widehat{\boldsymbol{q}}_{t+1})$ while the environment generates the next state s_{t+1} and reward r_t. We employ Proximal Policy Optimization (PPO) [245] to find the optimal policy $\boldsymbol{\pi}^*_{\text{UHC}}$ that maximizes the expected discounted return $\mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} r_t]$.

State. The state $\boldsymbol{s}_t \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$ of the humanoid contains the character's current pose $\boldsymbol{q}_t$ and joint velocity $\dot{\boldsymbol{q}}_t$. Here, the state $\boldsymbol{s}_t$ encapsulates the humanoid's full physical state at time step t. It only includes information about the current frame $(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$ and does not include any extra information, enabling our learned controller to be guided by a target pose only.

Action. The action $\boldsymbol{a}_t$ specifies the target joint angles for the proportional derivative (PD) controller [268] at each degree of freedom (DoF) of the humanoid joints except for the root (pelvis). We use the residual action representation [46]: $\boldsymbol{q}_t^d = \widehat{\boldsymbol{q}}_{t+1} + \boldsymbol{a}_t$, where $\boldsymbol{q}_t^d$ is the final PD target, $\boldsymbol{a}_t$ is the output of the control policy $\boldsymbol{\pi}_{\text{UHC}}$, and $\widehat{\boldsymbol{q}}_{t+1}$ is the target pose. The torque to be applied at joint i is: $\boldsymbol{\tau}^i = \boldsymbol{k}^p \circ (\boldsymbol{q}_t^d - \boldsymbol{q}_t) - \boldsymbol{k}^d \circ \dot{\boldsymbol{q}}_t$ where $\boldsymbol{k}^p$ and $\boldsymbol{k}^d$ are manually specified gains and $\circ$

is the element-wise multiplication. As observed in prior work [327,329], allowing the policy to apply external residual forces $\boldsymbol{\eta}_t$ to the root helps stabilizing the humanoid, so our final action is $\boldsymbol{a}_t \triangleq (\Delta\widetilde{\boldsymbol{q}}_t^d, \boldsymbol{\eta}_t)$.

Policy. The policy $\boldsymbol{\pi}_{\text{UHC}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \widehat{\boldsymbol{q}}_{t+1})$ is represented by a Gaussian distribution with a fixed diagonal covariance matrix Σ. We first use a feature extraction layer $D_{\text{diff}}(\widehat{\boldsymbol{q}}_{t+1}, \boldsymbol{q}_t)$ to compute the root and joint offset between the simulated pose and target pose. All features are then translated to a root-relative coordinate system using an agent-centic transform $\boldsymbol{T}_{\text{AC}}$ to make our policy orientation-invariant. We use a Multi-layer Perceptron (MLP) as our policy network to map the augmented state $\boldsymbol{T}_{\text{AC}}\left(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widehat{\boldsymbol{q}}_{t+1}, D_{\text{diff}}(\widehat{\boldsymbol{q}}_{t+1}, \boldsymbol{q}_t)\right)$ to the predicted action $\boldsymbol{a}_t$.

Reward function. For UHC, the reward function is designed to encourage the simulated pose $\boldsymbol{q}_t$ to better match the target pose $\widehat{\boldsymbol{q}}_{t+1}$. Since we share a similar objective (mimic target motion), our reward is similar to Residual Force Control [327].

Training procedure. We train our controller on the AMASS [176] dataset, which contains 11505 high-quality MoCap sequences with 4000k frame of poses (after removing sequences involving human-object interaction like running on a treadmill). At the beginning of each episode, a random fixed length sequence (300 frames) is sampled from the dataset for training. While prior works [292,297] uses more complex motion clustering techniques to sample motions, we devise a simple yet empirically effective sampling technique by inducing a probability distribution based on the value function. For each pose frame $\widehat{\boldsymbol{q}}_j$ in the dataset, we first compute an initialization state $\boldsymbol{s}_1^j$: $\boldsymbol{s}_1^j \triangleq (\widehat{\boldsymbol{q}}_j, \boldsymbol{0})$, and then score it using the value function to access how well the policy can mimic the sequence $\widehat{\boldsymbol{q}}_{j:T}$ starting from this pose: $V(\boldsymbol{s}_1^j, \widehat{\boldsymbol{q}}_{j+1}) = v_j$. Intuitively, the higher $\boldsymbol{v}_j$ is, the more confident our policy is in mimicing this sequence, and the less often we should pick this frame. The probability of choosing frame j, comparing against all frames J in the AMASS dataset, is then $P(\widehat{\boldsymbol{q}}_j) = \frac{\exp(-v_j/\tau)}{\sum_i^J \exp(-v_i/\tau)}$ where τ is the sampling temperature. More implementation details about the reward, training, and evaluation of UHC can be found in Section 2.5.8.

2.3.2 Kinematic Model – Object-aware Kinematic Policy

To leverage the power of our learned UHC, we design an auto-regressive and object-aware kinematic policy to generate per-frame target motion from egocentric inputs. We synchronize the state space of our kinematic policy and UHC such that the policy can be learned with or without physics simulation. When trained without physics simulation, the model is purely kinematic and can be optimized via supervised learning; when trained with a physics simulation, the model can be optimized through a combination of supervised learning and reinforcement learning. The latter procedure, coined *dynamics-regulated training*, enables our model to distill human dynamics information learned from large-scale MoCap data into the kinematic model and learns a policy more robust to convariate shifts. In this section, we will describe the architecture of the policy itself and the training through supervised learning (without physics simulation).

Scene context modelling and initialization. To serve as a high-level target motion estimator for egocentric videos with potential human-object interaction, our kinematic policy needs to be object-aware and grounded with visual input. To this end, given an input image sequence $\boldsymbol{I}_{1:T}$,

we compute the initial object states $\widetilde{o}_1$, camera trajectory $\widetilde{h}_{1:T}$ or image features $\phi_{1:T}$ as inputs to our system. The object states, $\widetilde{o}_t \triangleq (\widetilde{o}_t^{cls}, \widetilde{o}_t^{\text{pos}}, \widetilde{o}_t^{\text{rot}})$, is modeled as a vector concatenation of the main object-of-interest's class $\widetilde{o}_t^{cls}$, 3D position $\widetilde{o}_t^{\text{pos}}$, and rotation $\widetilde{o}_t^{\text{rot}}$. $\widetilde{o}_t$ is computed using an off-the-shelf object detector and pose estimator [111]. When there are no objects in the current scene (for walking and running *etc.*), the object states vector is set to zero. To provide our model with movement cues, we can either use image level optical flow feature $\phi_{1:T}$ or camera trajectory extracted from images using SLAM or VIO. Concretely, the image features $\phi_{1:T}$ is computed using an optical flow extractor [260] and ResNet [95]. The camera trajectory in our real-world experiments are computed using an off-the-shelf VIO method [110]: $\widetilde{h}_t \triangleq (\widetilde{h}_t^{\text{pos}}, \widetilde{h}_t^{\text{rot}})$ (position $\widetilde{h}_t^{\text{pos}}$ and orientation $\widetilde{h}_t^{\text{rot}}$). As visual input can be noisy, using the camera trajectory $\widetilde{h}_t$ directly significantly improves the performance of our framework as shown in our ablation studies (Sec. 2.4.2).

To provide our UHC with a plausible initial state for simulation, we estimate $\widetilde{q}_1$ from the scene context features $\widetilde{o}_{1:T}$ and $\phi_{1:T}$ / $\widetilde{h}_{1:T}$. We use an Gated Recurrent Unit (GRU) [49] based network to regress the initial agent pose $\widetilde{q}_1$. Combining the above procedures, we obtain the context modelling and initialization model $\pi_{\text{KIN}}^{\text{init}}$: $[\widetilde{q}_1, \widetilde{o}_1, \widetilde{h}_{1:T}/\phi_{1:T},] = \pi_{\text{KIN}}^{\text{init}}(\boldsymbol{I}_{1:T})$. Notice that to constrain the ill-posed problem of egocentric pose estimation, we assume known object category, rough size, and potential mode of interaction. We use these knowledge as a prior for our per-step model.

Training kinematic policy via supervised learning. After initialization, we use the estimated first-frame object pose $\widetilde{o}_t$ and human pose $\widetilde{q}_t$ to *initialize* the physics simulation. All subsequent object movements are a result of human-object interation and simulation. At each time step, we use a per-step model $\pi_{\text{KIN}}^{\text{step}}$ to compute the next frame pose based on the next frame observations: we obtain an egocentric input vector $\widetilde{c}_t$ through the agent-centric transformation function $\widetilde{c}_t = \boldsymbol{T}_{\text{AC}}(\widetilde{q}_t, \widetilde{o_{t+1}}, \widetilde{h}_{t+1}/\phi_{t+1})$ where $\tilde{c}_t \triangleq (\widetilde{r}_t'^{rot}, \widetilde{j}_t^{\text{rot}}, \widetilde{o'}_{t+1}, \widetilde{h}'_{t+1}/\phi_{t+1})$ contains the current agent-centric root orientation $\widetilde{r}_t'^{rot}$, joint angles $\widetilde{j}_t^{\text{rot}}$, and image feature for next frame ϕ_{t+1}, object state $\widetilde{o}'_{t+1}$ or camera pose $\widetilde{h}'_{t+1}$. From $\widetilde{c}_t$, the kinematic policy $\pi_{\text{KIN}}^{\text{step}}$ computes the root angular velocity $\widetilde{w}_t$, linear velocity $\widetilde{v}_t$, and next frame joint rotation $\widetilde{j}_{t+1}^{\text{rot}}$. The next frame pose is computed through a finite integration module $\boldsymbol{T}_{\text{finite}}$ with time difference $\delta t = 1/30s$:

$$\widetilde{\omega}_t, \widetilde{v}_t, \widetilde{j}_{t+1}^{\text{rot}} = \pi_{\text{KIN}}^{\text{step}}(\widetilde{c}_t), \quad \widetilde{q}_{t+1} = \boldsymbol{T}_{\text{finite}}(\widetilde{\omega}_t, \widetilde{v}_t, \widetilde{j}_{t+1}^{\text{rot}}, \widetilde{q}_t). \tag{2.1}$$

When trained without physics simulation, we auto-regressively apply the kinematic policy and use the computed $\widetilde{q}_{t+1}$ as the input for the next timestep. This procedure is outlined at Alg. 1. Since all mentioned calculations are end-to-end differentiable, we can directly optimize our $\pi_{\text{KIN}}^{\text{init}}$ and $\pi_{\text{KIN}}^{\text{step}}$ through supervised learning. Specifically, given ground truth $\widehat{q}_{1:T}$ and estimated $\widetilde{q}_{1:T}$ pose sequence, our loss is computed as the difference between the desired and ground truth values of the following quantities: agent root position ($\widehat{r}_t^{\text{pos}}$ vs $\widetilde{r}_t^{\text{pos}}$) and orientation ($\widehat{r}_t^{\text{rot}}$ vs $\widetilde{r}_t^{\text{rot}}$), agent-centric object position ($\widehat{o}_t'^{\text{pos}}$ vs $\widetilde{o}_t'^{\text{pos}}$) and orientation ($\widehat{o}_t'^{\text{rot}}$ vs $\widetilde{o}_t'^{\text{rot}}$), and agent joint orientation ($\widehat{j}_t^{\text{rot}}$ vs $\widetilde{j}_t^{\text{rot}}$) and position ($\widehat{j}_t^{\text{pos}}$ vs $\widetilde{j}_t^{\text{pos}}$, computed using forward kinematics):

$$\mathcal{L}_{\text{SL}} = \sum_{i=1}^{T} \|\widetilde{r}_t^{\text{rot}} \ominus \widehat{r}_t^{\text{rot}}\|^2 + \|\widetilde{r}_t^{\text{pos}} - \widehat{r}_t^{\text{pos}}\|^2 + \|\widetilde{o}_t'^{\text{rot}} \ominus \widehat{o}_t'^{\text{rot}}\|^2 + \|\widetilde{o}_t'^{\text{pos}} - \widehat{o}_t'^{\text{pos}}\|^2 + \|\widetilde{j}_t^{\text{rot}} \ominus \widehat{j}_t^{\text{rot}}\|^2 + \|\widetilde{j}_t^{\text{pos}} - \widehat{j}_t^{\text{pos}}\|^2. \tag{2.2}$$

Algo 1: Learning Kinematic Policy via Supervised Learning

Function `TrainSL`($\pi_{KIN}^{init}, \pi_{KIN}^{step}, \boldsymbol{I}, \widehat{\boldsymbol{Q}}$)**:**
  while *not converged* **do**
    $\boldsymbol{M}_{\text{SL}} \leftarrow \emptyset$ `// initialize sampling memory;`
    while $\boldsymbol{M}_{SL}$ *not full* **do**
      $\boldsymbol{I}_{1:T} \leftarrow$ random sequence from dataset $\boldsymbol{I}$;
      $\widetilde{\boldsymbol{q}}_1, \widetilde{\boldsymbol{o}}_1, \widetilde{\boldsymbol{h}}_{1:T}/\boldsymbol{\phi}_{1:T} \leftarrow \boldsymbol{\pi}_{\text{KIN}}^{\text{init}}(\boldsymbol{I}_{1:T})$ `// compute scene context and initial pose;`
      for $t \leftarrow 1...T$ **do**
        $\widetilde{\boldsymbol{c}}_t \leftarrow \boldsymbol{T}_{\text{AC}}(\widetilde{\boldsymbol{q}}_t, \widetilde{\boldsymbol{o}}_{t+1}, \widetilde{\boldsymbol{h}}_{t+1}/\boldsymbol{\phi}_{t+1})$ `// compute agent-centric input features;`
        $\widetilde{\boldsymbol{q}}_{t+1} \leftarrow \boldsymbol{T}_{\text{finite}}(\boldsymbol{\pi}_{\text{KIN}}^{\text{step}}(\widetilde{\boldsymbol{c}}_t), \widetilde{\boldsymbol{q}}_t)$;
        store $(\widetilde{\boldsymbol{q}}_t, \widehat{\boldsymbol{q}}_t)$ into memory $\boldsymbol{M}_{\text{SL}}$;
    $\boldsymbol{\pi}_{\text{KIN}}^{\text{step}}, \boldsymbol{\pi}_{\text{KIN}}^{\text{init}} \leftarrow$ supervised learning update using data in $\boldsymbol{M}_{\text{SL}}$ for 10 epochs;
  return $\boldsymbol{\pi}_{\text{KIN}}^{\text{init}}, \boldsymbol{\pi}_{\text{KIN}}^{\text{step}}$;

Input: Egocentric video dataset $\boldsymbol{I}$ and ground truth motion dataset $\widehat{\boldsymbol{Q}}$;
Initialize $\boldsymbol{\pi}_{\text{KIN}}^{\text{init}}, \boldsymbol{\pi}_{\text{KIN}}^{\text{step}}$;
`TrainSL`($\pi_{KIN}^{init}, \pi_{KIN}^{step}, \boldsymbol{I}, \widehat{\boldsymbol{Q}}$)

2.3.3 dynamics-Regulated Training

To tightly integrate our kinematic and dynamics models, we design a *dynamics-regulated training* procedure, where the kinematic policy learns from explicit physics simulation. In the procedure described in the previous section, the next-frame pose fed into the network is computed through finite integration and is not checked by physical laws: whether a real human can perform the computed pose is never verified. Intuitively, this amounts to mentally think about moving in a physical space *without actually moving*. Combining our UHC and our kinematic policy, we can leverage the prelearned motor skills from UHC and let the kinematic policy act directly in a simulated physical space to obtain feedback about physical plausibility. The procedure for dynamics-regulated training is outlined in Alg. 2. In each episode, we use $\boldsymbol{\pi}_{\text{KIN}}^{\text{init}}$ and $\boldsymbol{\pi}_{\text{KIN}}^{\text{step}}$ as in Alg. 1, with the key distinction being: at the next timestep $t+1$, the input to the kinematic policy is the result of UHC and physics simulation $\boldsymbol{q}_{t+1}$ instead of $\widetilde{\boldsymbol{q}}_{t+1}$. $\boldsymbol{q}_{t+1}$ explicitly verify that the $\widetilde{\boldsymbol{q}}_{t+1}$ produced by the kinematic policy can be successfully followed by a motion controller. Using $\boldsymbol{q}_{t+1}$ also informs our $\boldsymbol{\pi}_{\text{KIN}}^{\text{step}}$ of the current humanoid state and encourages the policy to adjust its predictions to improve humanoid stability.

dynamics-regulated optimization. Since the physics simulation is not differentiable, we cannot directly optimize the simulated pose $\boldsymbol{q}_t$; however, we can optimize $\boldsymbol{q}_t$ through reinforcement learning and $\widetilde{\boldsymbol{q}}_t$ through supervised learning. Since we know that $\widehat{\boldsymbol{q}}_t$ is a *good guess* reference motion for UHC, we can directly optimize $\widetilde{\boldsymbol{q}}_t$ via supervised learning as done in Sec. 2.3.2 using the loss defined in Eq. 2.2. Since the data samples are collected through physics simulation, the input $\boldsymbol{q}_t$ is physically-plausible and more diverse than those collected purely through auto-regressively applying $\boldsymbol{\pi}_{\text{KIN}}^{\text{step}}$ in Alg. 1. This way, our dynamics-regulated training procedure

Algo 2: Learning Kinematic Policy via dynamics-Regulated Training

Function `TraindynamicsRegulated`($\pi_{KIN}^{init}, \pi_{KIN}^{step}, \pi_{UHC}, \boldsymbol{I}, \widehat{\boldsymbol{Q}}$)**:**
  Train $\pi_{\text{KIN}}^{\text{init}}, \pi_{\text{KIN}}^{\text{step}}$ using Algorithm 1 for 20 epochs (optional) ;
  while *not converged* **do**
    $\boldsymbol{M}_{\text{dyna}} \leftarrow \emptyset$ `// initialize sampling memory` ;
    while $\boldsymbol{M}_{dyna}$ *not full* **do**
      $\boldsymbol{I}_{1:T} \leftarrow$ random sequence from dataset $\boldsymbol{I}$;
      $\boldsymbol{q}_1 \leftarrow \widetilde{\boldsymbol{q}}_1, \widetilde{\boldsymbol{o}}_1, \widetilde{\boldsymbol{h}}_{1:T}/\boldsymbol{\phi}_{1:T} \leftarrow \pi_{\text{KIN}}^{\text{init}}(\boldsymbol{I}_{1:T})$ `// compute scene context and initial pose`;
      $\boldsymbol{s}_1 \leftarrow (\boldsymbol{q}_1, \dot{\boldsymbol{q}}_1)$ `// initialize simulation state`;
      for $t \leftarrow 1...T$ **do**
        $\boldsymbol{c}_t \leftarrow \boldsymbol{T}_{\text{AC}}(\boldsymbol{q}_t, \widetilde{\boldsymbol{o}}_{t+1}, \widetilde{\boldsymbol{h}}_{t+1}/\boldsymbol{\phi}_{t+1})$ `// agent-centric features`;
        $\widetilde{\boldsymbol{q}}_{t+1} \sim \boldsymbol{T}_{\text{finite}}(\pi_{\text{KIN}}^{\text{step}}(\boldsymbol{c}_t), \boldsymbol{q}_t)$ `// sample next pose`;
        $\boldsymbol{s}_t \leftarrow (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$;
        $\boldsymbol{a}_t \leftarrow \pi_{\text{UHC}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \widetilde{\boldsymbol{q}}_{t+1})$;
        $\boldsymbol{s}_{t+1} \leftarrow P_{\text{physics}}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulate physics step`;
        $\boldsymbol{q}_{t+1} \leftarrow \boldsymbol{s}_{t+1}, \boldsymbol{r}_t^{\text{KIN}} \leftarrow$ reward from Eq. 2.3 ;
        store $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1}, \widehat{\boldsymbol{q}}_t, \widetilde{\boldsymbol{q}}_{t+1})$ into memory $\boldsymbol{M}_{\text{dyna}}$;
    Update $\pi_{\text{KIN}}^{\text{step}}$ with reinforcement learning using data in $\boldsymbol{M}_{\text{dyna}}$ for 10 epochs ;
    Update $\pi_{\text{KIN}}^{\text{init}}, \pi_{\text{KIN}}^{\text{step}}$ with supervised learning using data in $\boldsymbol{M}_{\text{dyna}}$ for 10 epochs ;
  return $\pi_{\text{KIN}}^{\text{init}}, \pi_{\text{KIN}}^{\text{step}}$;

Input: Pre-trained controller π_{UHC}, egocentric video dataset $\boldsymbol{I}$, and ground truth motion dataset $\widehat{\boldsymbol{Q}}$;
Initialize $\pi_{\text{KIN}}^{\text{init}}, \pi_{\text{KIN}}^{\text{step}}$;
`TraindynamicsRegulated`($\pi_{KIN}^{init}, \pi_{KIN}^{step}, \pi_{UHC}, \boldsymbol{I}, \widehat{\boldsymbol{Q}}$)

performs a powerful data augmentation step, exposing $\pi_{\text{KIN}}^{\text{step}}$ with diverse states collected from simulation.

However, MoCap pose $\widehat{\boldsymbol{q}}_t$ is imperfect and can contain physical violations itself (foot-skating, penetration *etc.*), so asking the policy $\pi_{\text{KIN}}^{\text{step}}$ to produce $\widehat{\boldsymbol{q}}_t$ as reference motion *regardless of the current humanoid state* can lead to instability and cause the humanoid to fall. The kinematic policy should adapt to the current simulation state and provide reference motion $\widetilde{\boldsymbol{q}}_t$ that can lead to poses similar to $\widehat{\boldsymbol{q}}_t$ yet still physically-plausible. Such behavior will not emerge through supervised learning and require *trial and error*. Thus, we optimize $\pi_{\text{KIN}}^{\text{step}}$ through reinforcement learning and reward maximization. We design our RL reward to have two components: motion imitation and dynamics self-supervision. The motion imitation reward encourages the policy to match the computed camera trajectory $\widetilde{\boldsymbol{h}}_t$ and MoCap pose $\widehat{\boldsymbol{q}}_t$, and serves as a regularization on motion imitation quality. The dynamics self-supervision reward is based on the insight that the disagreement between $\widetilde{\boldsymbol{q}}_t$ and $\boldsymbol{q}_t$ contains important information about the quality and physical plausibility of $\widetilde{\boldsymbol{q}}_t$: the better $\widetilde{\boldsymbol{q}}_t$ is, the easier it should be for UHC to mimic it. Formally, we define

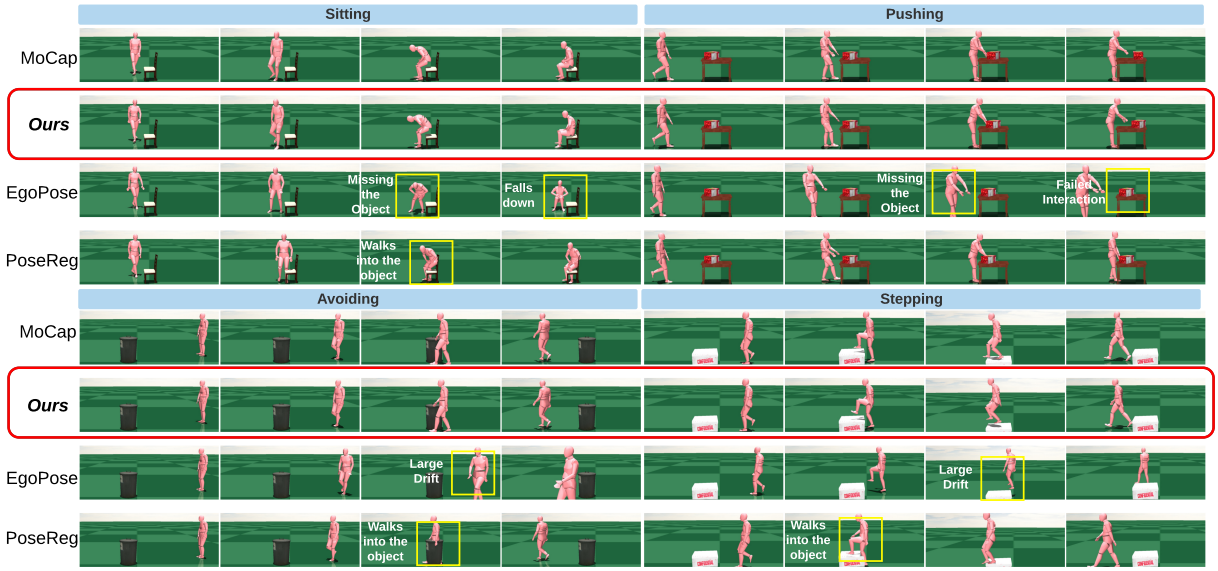


Figure 2.3. Results of egocentric pose and human-object interaction estimation from the MoCap datset.

the reward for $\boldsymbol{\pi}^{\text{step}}_{\text{KIN}}$ as:

$$\begin{aligned}\boldsymbol{r}_t = w_{\text{hp}} e^{-45.0(\|\boldsymbol{h}^{\text{pos}}{}_t \ominus \widetilde{\boldsymbol{h}}^{\text{pos}}_t\|^2)} + w_{\text{hq}} e^{-45.0(\|\boldsymbol{h}^{\text{rot}}{}_t \ominus \widetilde{\boldsymbol{h}}^{\text{rot}}_t\|^2)} + w^{\text{gt}}_{\text{jv}} e^{-0.005(\|\dot{\boldsymbol{j}}^{\text{rot}}_t \ominus \widehat{\dot{\boldsymbol{j}}^{\text{rot}}_t}\|^2)} + \\ w^{\text{gt}}_{\text{jr}} e^{-50.0(\|\boldsymbol{j}^{\text{rot}}_t \ominus \widehat{\boldsymbol{j}}^{\text{rot}}_t\|^2)} + w^{\text{dyna}}_{\text{jr}} e^{-50.0(\|\boldsymbol{j}^{\text{rot}} \ominus \widetilde{\boldsymbol{j}}^{\text{rot}}_t\|^2)} + w^{\text{dyna}}_{\text{jp}} e^{-50.0(\|\boldsymbol{j}^{\text{pos}}_t - \widetilde{\boldsymbol{j}}^{\text{pos}}_t\|^2)},\end{aligned} \tag{2.3}$$

w_{hp}, w_{hq} are weights for matching the extracted camera position $\widetilde{\boldsymbol{h}}^{\text{pos}}_t$ and orientation $\widetilde{\boldsymbol{h}}^{\text{rot}}_t$; $w^{\text{gt}}_{jr}, w^{\text{gt}}_{\text{jv}}$ are for matching ground truth joint angles $\widehat{\boldsymbol{j}}^{\text{rot}}$ and angular velocities $\widehat{\dot{\boldsymbol{j}}^{\text{rot}}_t}$. $w^{\text{dyna}}_{\text{jr}}$, $w^{\text{dyna}}_{\text{jp}}$ are weights for the dynamics self-supervision rewards, encouraging the policy to match the target kinematic joint angles $\widetilde{\boldsymbol{j}}^{\text{rot}}_t$ and positions $\widetilde{\boldsymbol{j}}^{\text{pos}}_t$ to the simulated joint angles $\boldsymbol{j}^{\text{rot}}_t$ and positions $\boldsymbol{j}^{\text{pos}}_t$. As demonstrated in Sec. 2.4.2, the RL loss is particularly helpful in adapting to challenging real-world sequences, which requires the model to adjust to domain shifts and unseen motion.

Test-time. At the test time, we follow the same procedure outlined in Alg 2 and Fig.2.2 to roll out our policy to obtain simulated pose $\boldsymbol{q}_{1:T}$ given a sequence of images $\boldsymbol{I}_{1:T}$. The difference being instead of sampling from $\boldsymbol{\pi}^{\text{step}}_{\text{KIN}}(\boldsymbol{c}_t)$ as a Guassian policy, we use the mean action directly.

2.4 Experiments

Datasets. As no public dataset contains synchronized ground-truth full-body pose, object pose, and egocentric videos with human-object interactions, we record two egocentric datasets: one inside a MoCap studio, another in the real-world. The MoCap dataset contains 266 sequences (148k frames) of paired egocentric videos and annotated poses. It features one of the five actions: sitting down on a chair, avoiding obstacles, stepping on a box, pushing a box, and generic locomotion [324] (walking, running, crouching) recorded using a head-mounted GoPro. Each action has around 50 sequences with different starting position and facing, gait, speed *etc*. We use an 80–20 train test data split on this MoCap dataset. The real-world dataset is only for

testing purpose and contains 183 sequences (50k frames) of an additional subject performing similar actions in an everyday setting wearing a head-mounted iPhone. For both datasets, we use different objects and varies the object 6DoF pose for each capture take. Additional details (diversity, setup *etc.*) can be found in Section 2.5.12.

Evaluation metrics. We use both pose-based and physics-based metrics for evaluation. To evaluate the 3D global pose accuracy, we report the root pose error (E_{root}) and root-relative mean per joint position error [132] (E_{mpjpe}). When ground-truth root/pose information is unavailable (for real-world dataset), we substitute E_{root} with E_{cam} to report camera pose tracking error. We also employ four physics based pose metrics: acceleration error (E_{acc}), foot skating (FS), penetration (PT), and interaction success rate (S_{inter}). E_{acc} (mm/frame^2) compares the ground truth and estimated average joint acceleration; FS (mm) is defined the same as in Ling *et al.* [153]; PT (mm) measures the average penetration distance between our humanoid and the scene (ground floor and objects). Notice that our MoCap dataset has *an penetration of 7.182 mm and foot sliding of 2.035 mm* per frame, demonstrating that the MoCap data is imperfect and may not serve as the best target motion. S_{inter} is defined as whether the objects of interest has been moved enough (pushing and avoiding) or if desired motion is completed (stepping and sitting). If the humanoid falls down at any point, $S_{inter} = 0$. For a full definition of our evaluation metrics, please refer to Section 2.5.2.

Baseline methods. To show the effectiveness of our framework, we compare with the previous state-of-the-art egocentric pose estimation methods: (1) the best dynamics-based approach *EgoPose* [325] and (2) the best kinematics-based approach *PoseReg*, also proposed in [325]. We use the official implementation and augment their input with additional information ($\widetilde{\boldsymbol{o}}_t$ or $\widetilde{\boldsymbol{h}}_t$) for a fair comparison. In addition, we incorporate the fail-safe mechanism [325] to reset the simulation when the humanoid loses balance to ensure the completion of each sequence (details in Section 2.5.2).

Implementation details. We use the MuJoCo free physics simulator [279] and run the simulation at 450 Hz. Our learned policy is run every 15 timesteps and assumes that all visual inputs are at 30 Hz. The humanoid follows the kinematic and mesh definition of the SMPL model and has 25 bones and 76 DoF. We train our method and baselines on the training split (202 sequences) of our MoCap dataset. The training process takes about 1 day on an RTX 2080-Ti with 35 CPU threads. After training and the initialization step, our network is causal and runs at 50 FPS on an Intel desktop CPU. The main evaluation is conducted using head poses, and we show results on using the image features in ablation. For more details on the implementation, refer to Section 2.5.2 and Section 2.5.8.

2.4.1 Results

MoCap dataset results. Table 2.1 shows the quantitative comparison of our method with the baselines. All results are averaged across five actions, and all models have access to the same inputs. We observe that our method, trained either with supervised learning or dynamics-regulated, outperform the two state-of-the-art methods across all metrics. Not surprisingly, our purely kinematic model performs the best on pose-based metrics, while our dynamics-regulated trained policy excels at the physics-based metrics. Comparing the kinematics-only models we

MoCap dataset							
Method	Physics	S_{inter} ↑	E_{root} ↓	E_{mpjpe} ↓	E_{acc} ↓	FS ↓	PT ↓
PoseReg	✗	-	0.857	87.680	12.981	8.566	42.153
Kin_poly: supervised learning (ours)	✗	-	**0.176**	**33.149**	**6.257**	5.579	10.076
EgoPose	✓	48.4%	1.957	139.312	9.933	**2.566**	7.102
Kin_poly: dynamics-regulated (ours)	✓	**96.9%**	0.204	39.575	6.390	3.075	**0.679**

Real-world dataset									
Method	Physics	S_{inter} ↑	E_{cam} ↓	FS ↓	PT ↓	Per class success rate S_{inter} ↑			
PoseReg	✗	-	1.260	6.181	50.414	Sit	Push	Avoid	Step
Kin_poly: supervised learning (ours)	✗	-	0.491	5.051	34.930				
EgoPose	✓	9.3%	1.896	**2.700**	1.922	7.93%	6.81%	4.87%	0.2%
Kin_poly: dynamics-regulated (ours)	✓	**93.4%**	**0.475**	2.726	**1.234**	**98.4%**	**95.4%**	**100%**	**74.2%**

Table 2.1. Quantitative results on pose and physics based metrics on the MoCap and real-world Dataset.

can see that our method has a much lower (79.4% error reduction) root and joint position error (62.1% error reduction) than PoseReg, which shows that our object-aware and autoregressive design of the kinematic model can better utilize the provided visual and scene context and avoid compounding errors. Comparing with the dynamics-based methods, we find that the humanoid controlled by EgoPose has a much larger root drift, often falls down to the ground, and has a much lower success rate in human-object interaction (48.4 % vs 96.9%). Upon visual inspection in Fig. 2.3, we can see that our kinematic policy can faithfully produce human-object interaction on almost every test sequence from our MoCap dataset, while PoseReg and EgoPose often miss the object-of-interest (as can be reflected by the large root tracking error). Both of the dynamics-based methods has smaller acceleration error, foot skating, and penetration; some even smaller than MoCap (which has 2 mm FS and 7mm PT). Notice that our joint position error is relatively low compared to state-of-the-art third-person pose estimation methods [132, 135, 171] due to our strong assumption about known object of interest, its class, and potential human-object interactions, which constrains the ill-posed problem pose estimation from just front-facing cameras.

Real-world dataset results. The real-world dataset is far more challenging, having similar number of sequences (183 clips) as our training set (202 clips) and recorded using different equipment, environments, and motion patterns. Since no ground-truth 3D poses are available, we report our results on camera tracking and physics-based metrics. As shown in Table 2.1, our method outperforms the baseline methods by a large margin in almost all metrics: although EgoPose has less foot-skating (as it also utilizes a physics simulator), its human-object interaction success rate is extremely low. This can be also be reflected by the large camera trajectory error, indicating that the humanoid is drifting far away from the objects. The large drift can be attributed to the domain shift and challenging locomotion from the real-world dataset, causing EgoPose's humanoid controller to accumulate error and lose balance easily. On the other hand, our method is able to generalize and perform successful human-object interactions, benefiting from our pretrained UHC and kinematic policy's ability to adapt to new domains and motion.

Table 2.1 also shows a success rate breakdown by action. Here we can see that "stepping on a box" is the most challenging action as it requires the humanoid lifting its feet at a precise moment and pushing itself up. Note that our UHC has never been trained on any stepping or human-object interaction actions (as AMASS has no annotated object pose) but is able to perform these action. As motion is best seen in videos, we refer readers to our supplementary video.

2.4.2 Ablation Study

To evaluate the importance of our components, we train our kinematic policy under different configurations and study its effects on the *real-world dataset*, which is much harder than the MoCap dataset. The results are summarized in Table 2.2. Row 1 (R1) corresponds to training the kinematic policy only with Alg. 1 only and use UHC to mimic the target kinematic motion as a post-processing step. Row 2 (R2) are the results of using dynamics-regulated training but only performs the supervised learning part. R3 show a model trained with optical flow image features rather than the estimated camera pose from VIO. Comparing R1 and R2, the lower interaction success rate (73.2% vs 80.9%) indicates that exposing the kinematic policy to states from the physics simulation serves as a powerful data augmentation step and leads to a model more robust to real-world scenarios. R2 and R4 show the benefit of the RL loss in dynamics-regulated training: allowing the kinematic policy to deviate from the MoCap poses makes the model more adaptive and achieves higher success rate. R3 and R4 demonstrate the importance of *intelligently* incorporating extracted camera pose as input: visual features ϕ_t can be noisy and suffer from domain shifts, and using techniques such as SLAM and VIO to extract camera poses as an additional input modality can largely reduce the root drift. Intuitively, the image features computed from optical flow and the camera pose extracted using VIO provide a similar set of information, while VIO provides a cleaner information extraction process. Note that our kinematic policy *without using* extracted camera trajectory outperforms EgoPose that *uses camera pose* in both success rate and camera trajectory tracking. Upon visual inspection, the humanoid in R3 largely does not fall down (compared to EgoPose) and mainly attributes the failure cases to drifting too far from the object.

Component				Metric			
SL	Dyna_reg	RL	VIO	S_{inter} ↑	E_{cam} ↓	FS ↓	PT ↓
✓	✗	✗	✓	73.2%	0.611	4.234	1.986
✓	✓	✗	✓	80.9%	0.566	3.667	4.490
✓	✓	✓	✗	54.1%	1.129	7.070	5.346
✓	✓	✓	✓	**93.4%**	**0.475**	**2.726**	**1.234**

Table 2.2. Ablation study of different components of our framework.

2.5 Additional Details

2.5.1 Qualitative Results (Supplemantry Video)

As motion is best seen in videos, we provide extensive qualitative evaluations in the supplementary video. Here we list a timestamp reference for evaluations conducted in the video:

- Qualitative results from real-world videos (00:12).

- Comparison with the state-of-the-art methods on the MoCap dataset's test split (01:27).
- Comparison with the state-of-the-art methods on the real-world dataset (02:50).
- Failure cases for the dynamics-regulated kinematic policy (04:23).
- Qualitative results from the Universal Humanoid Controller (UHC) (4:39).
- Failure cases for the UHC (05:20).

2.5.2 dynamics-regulated Kinematic Policy

2.5.3 Evaluation Metrics Definition

Here we provide details about our proposed evaluation metrics:

- Root error: $\mathbf{E}_{\text{root}}$ compares the estimated and ground truth root rotation and orientation, measuring the difference in the respective 4×4 transformation matrix ($\boldsymbol{M}_t$): $\frac{1}{T}\sum_{t=1}^{T} \|I - (\boldsymbol{M}_t \widehat{\boldsymbol{M}}_t^{-1})\|_F$. This metric reflects both the position and orientation tracking quality.
- Mean per joint position error: $\mathbf{E}_{\text{mpjpe}}$ (mm) is the popular 3D human pose metric [125, 132,135] and is defined as $\frac{1}{J}\|\boldsymbol{j}^{\text{pos}} - \widehat{\boldsymbol{j}}^{\text{pos}}\|_2$ for J number of joints. This value is root-relative and is computed after setting the root translation to zero.
- Acceleration error: E_{acc} (mm/frame^2) measures the difference between the ground truth and estimated joint position acceleration: $\frac{1}{J}\|\ddot{\boldsymbol{j}}^{\text{pos}} - \widehat{\ddot{\boldsymbol{j}}}^{\text{pos}}\|_2$.
- Foot sliding: FS (mm) is computed similarly as in [153], *i.e.* FS $= d\left(2 - 2^{h/H}\right)$ where d is the foot displacement and h is the foot height of two consecutive poses. We use a height threshold of $H = 33$ mm, the same as in [153].
- Penetration: PT (mm) is provided by the physics simulation. It measures the per-frame average penetration distance between our simulated humanoid and the scene (ground and objects). Notice that Mujoco uses a soft contact model where a larger penetration will result in a larger repulsion force, so a small amount of penetration is expected.
- Camera trajectory error: $\mathbf{E}_{\text{cam}}$ is defined the same as the root error, and measures the camera trajectory tracking instead of the root. To extract the camera trajectory from the estimated pose $\boldsymbol{q}_t$, we use the head pose of the humanoid and apply a delta transformation based on the camera mount's vertical and horizontal displacement from the head.
- Human-object interaction success rate: S_{inter} measures whether the desired human-object interaction is successful. If the humanoid falls down at any point during the sequence, the sequence is deemed unsuccessful. The success rate is measured automatically by querying the position, contact, and simulation states of the objects and humanoid. For each action:
  - Sitting down: successful if the humanoid's pelvis or the roots of both legs come in contact with the chair at any point in time.

- Pushing a box: successful if the box is moved more than 10 cm during the sequence.
- Stepping on a box: successful if the humanoid's root is raised at least 10 cm off the ground and either foot of the humanoid has come in contact with the box.
- Avoiding an obstacle: successful if the humanoid has not come in contact with the obstacle and the ending position of the root/camera is less than 50 cm away from the desired position (to make sure the humanoid does not drift far away from the obstacle).

2.5.4 Fail-safe during evaluation

For methods that involve dynamics, the humanoid may fall down mid-episode and not complete the full sequence. In order to compare all methods fairly, we incorporate the "fail-safe" mechanism proposed in EgoPose [325] and use the estimated kinematic pose to restart the simulation at the timestep of failure. Concretely, we measure point of failure by thresholding the difference between the reference joint position $\tilde{\boldsymbol{j}}_t^{\text{pos}}$ and the simulated joint position $\boldsymbol{j}_t^{\text{pos}}$. To reset the simulation, we use the estimated kinematic pose $\tilde{\boldsymbol{q}}_t$ to set the simulation state.

2.5.5 Implementation Details

The kinematic policy is implemented as a Gated Recurrent Unit (GRU) [49] based network with 1024 hidden units, followed by a three-layer MLP (1024, 512, 256) with ReLU activation. The value function for training the kinematic policy through reinforcement learning is a two-layer MLP (512, 256) with ReLU activation. We use a fixed diagonal covariance matrix and train for 1000 epoches using the Adam [128] optimizer. Hyperparameters for training can be found in Table. 2.3:

	γ	Batch Size	Value Learning Rate	Policy Learning Rate	PPO clip ϵ	Covariance Std
Value	0.95	10000	3×10^{-4}	5×10^{-4}	0.2	0.04
	w_{hp}	w_{hq}	$w_{\text{jr}}^{\text{gt}}$	$w_{\text{jv}}^{\text{gt}}$	$w_{\text{jr}}^{\text{dyna}}$	$w_{\text{jp}}^{\text{dyna}}$
Value	0.15	0.15	0.2	0.1	0.2	0.2

Table 2.3. Hyperparameters used for training the kinematic policy.

2.5.6 Additional Experiments about Stochasticity

Our kinematic policy is trained through physics simulation and samples a random sequence from the MoCap dataset for each episode. Here we study the stochasticity that rises from this process. We train our full pipeline with three different random seeds and report its results with error bars on both the MoCap test split and the real-world dataset. As can be seen in Table 2.4, our method has very small stochasticity and maintains high performance on *both* the MoCap test split and the real-world dataset, demonstrating the robustness of our dynamics-regulated kinematic policy. Across different random seeds, we can see that "stepping" is consistently the hardest action and "avoiding" is the easiest. Intuitively, "stepping" requires precise coordination

MoCap dataset										
$S_{inter} \uparrow$	$E_{root} \downarrow$	$E_{mpjpe} \downarrow$	$E_{acc} \downarrow$	FS $\downarrow$	PT $\downarrow$	Per class success rate $S_{inter} \uparrow$				
						Sit	Push	Avoid	Step	Loco
$96.87\% \pm 1.27\%$	0.21 ± 0.01	39.46 ± 0.52	6.27 ± 0.1	3.22 ± 0.11	0.69 ± 0.03	100%	$97.20\% \pm 3.96\%$	100%	$86.70\% \pm 4.71\%$	$97.4\% \pm 3.63\%$

Real-world dataset							
$S_{inter} \uparrow$	$E_{root} \downarrow$	FS $\downarrow$	PT $\downarrow$	Per class success rate $S_{inter} \uparrow$			
				Sit	Push	Avoid	Step
$92.17\% \pm 1.41\%$	0.49 ± 0.01	2.72 ± 0.03	1.03 ± 0.16	$94.7\% \pm 4.20\%$	$93.10\% \pm 1.84\%$	100.0%	$77.1\% \pm 2.37\%$

Table 2.4. Results of our dynamics-regulated kinematic policy on the test split of MoCap and real-world datasets using different random seeds. The "loco" motion in the MoCap dataset corresponds to the generic locomotion action, containing all sequences from the EgoPose [325] Dataset.

between the kinematic policy and the UHC for lifting the feet and pushing up, while "avoiding" only requires basic locomotion skills.

2.5.7 Additional Analysis into low Per Joint Error

As discussed in the results section, we notice that our Mean Per Joint Position Error is relatively low compared to third-person pose estimation methods, although egocentric pose estimation is arguably a more ill-posed task. To provide an additional analysis of this observation, here we report the per-joint positional errors for the four joints with the smallest and largest errors, in ascending order in Table 2.5.

Torso	Left_hip	Right_hip	Spine	Left_toe	Right_toe	Right_hand	Left_hand
7.099	8.064	8.380	15.167	65.060	66.765	74.599	77.669

Table 2.5. Per-joint error on the MoCap dataset

As can be seen in the results, the toes and hands have much larger errors. This is expected as inferring hand and toe movements from only the egocentric view is challenging, and our network is able to extrapolate their position based on physical laws and prior knowledge of the scene context. Different from a third-person pose estimation setting, correctly estimating the torso area can be much easier from an egocentric point of view since torso movement is highly correlated with head motion. In summary, the low MPJPE reported on our MoCap dataset is the result of 1) only modeling a subset of possible human actions and human-object interactions, 2) the nature of the egocentric pose estimation task, 3) our network's incorporation of physical laws and scene context, which reduces the number of possible trajectories.

2.5.8 Universal Humanoid Controller

2.5.9 Implementation Details

Proxy humanoid. The proxy humanoid we use for simulation is created automatically using the mesh, bone and kinematic tree defined in the popular SMPL [161] human model. Similar to the procedure in [329], given the SMPL body vertices $V = 6890$ and bones $B = 25$, we generate

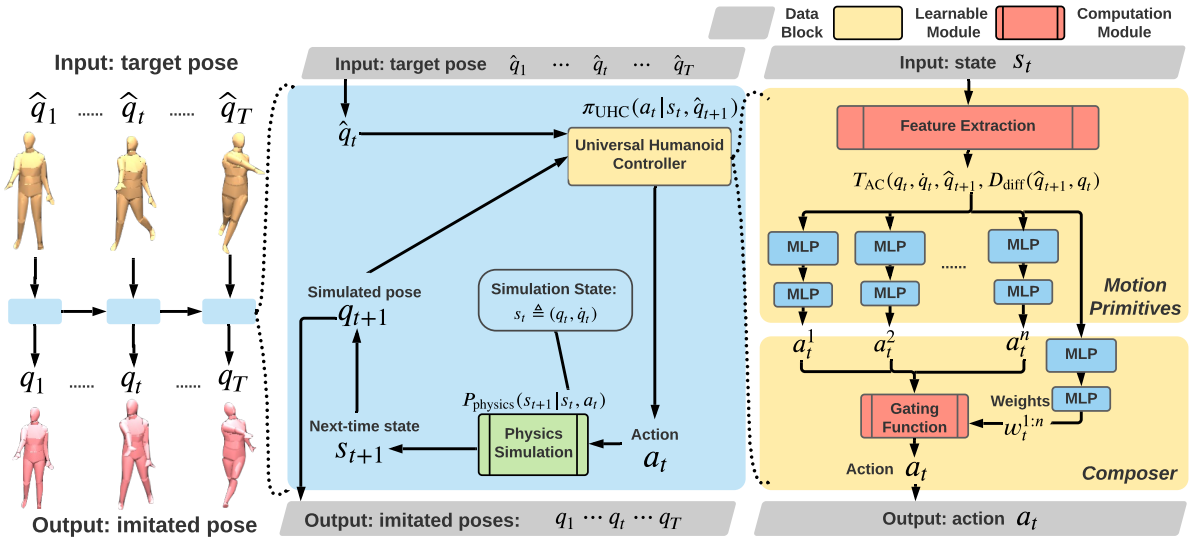


Figure 2.4. Overview of our Universal dynamics Controller. Given a frame of target pose and current simulation state, our UHC π_{UHC} can dirve the the humanoid to match the target pose.

our humanoid based on the skinning weight matrix $\boldsymbol{W} \in \mathbb{R}^{V \times B}$ that defines the association between each vertex and bone. The geometry of each bone's mesh is defined by the convex hull of all vertices assigned to the bone. The mass of each bone is in turn defined by the volume of the mesh. To simplify the simulation process, we discard all body shape information from the AMASS [176] dataset, and use the mean body shape of the SMPL model. Since AMASS and our MoCap dataset are recorded by people with different height, we manually adjust the starting height of the MoCap pose to make sure each of the humanoid's feet are touching the ground at the starting point of the episode.

Policy network architecture. Our Universal Humanoid Controller (UHC)'s workflow and architecture can be seen in Fig. 2.4. $\boldsymbol{\pi}_{\text{UHC}}(\boldsymbol{a}_t|\boldsymbol{s}_t, \widehat{\boldsymbol{q}}_{t+1})$ is implemented as a multiplicative compositional policy (MCP) [212] with eight motion primitives, each being an MLP with two hidden layers (512, 256). The composer is another MLP with two hidden layers (300, 200) and outputs the multiplicative weights $\boldsymbol{w}_t^{1:n}$ for the n motion primitives. As studied in MCP [212], this hierarchical control policy increases the model's capacity to learn multiple skills simultaneously. The output $\boldsymbol{a}_t \in \mathbb{R}^{75}$ is a vector concatenation of the target angles of the PD controller mounted on the 23 no-root joints (each has 3 DoF), plus the residual force [327]: $\boldsymbol{\eta}_t \in \mathbb{R}^6$. Recall that each target pose $\widehat{\boldsymbol{q}}_t \in \mathbb{R}^{76}$, $\widehat{\boldsymbol{q}}_t \triangleq (\widehat{\boldsymbol{r}}_t^{\text{pos}}, \widehat{\boldsymbol{r}}_t^{\text{rot}}, \widehat{\boldsymbol{j}}_t^{\text{rot}})$ consists of the root position $\widehat{\boldsymbol{r}}_t^{\text{pos}} \in \mathbb{R}^3$, root orientation in quaternions $\widehat{\boldsymbol{r}}_t^{\text{rot}} \in \mathbb{R}^4$, and body joint angles in Euler angles $\widehat{\boldsymbol{j}}_t^{\text{rot}} \in \mathbb{R}^{69}$ of the human model. The use of quaternions and Euler angles follows the specification of Mujoco [279]. As described in the main paper, our UHC first transforms the simulation state to a feature vector using $\boldsymbol{T}_{\text{AC}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widehat{\boldsymbol{q}}_{t+1}, D_{\text{diff}}(\widehat{\boldsymbol{q}}_{t+1}, \boldsymbol{q}_t))$ to output a 640 dimensional vector that is a concatenation of the following values:

$$(\boldsymbol{h}_t'^q, \boldsymbol{q}_t', \widehat{\boldsymbol{q}}_t', (\boldsymbol{q}_t - \widehat{\boldsymbol{q}}_t), \dot{\boldsymbol{q}}_t, (\psi_t - \widehat{\psi}_t), \widehat{\boldsymbol{j}}_t'^{\text{pos}}, (\boldsymbol{j}_t'^{\text{pos}} - \widehat{\boldsymbol{j}}_t'^{\text{pos}}), \boldsymbol{j}_t'^{\text{rot}}, (\boldsymbol{j}_t'^{\text{rot}} \ominus \widehat{\boldsymbol{j}}_t'^{\text{rot}})) = \boldsymbol{T}_{\text{AC}}(\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \widehat{\boldsymbol{q}}_{t+1}, D_{\text{diff}}(\widehat{\boldsymbol{q}}_{t+1}, \boldsymbol{q}_t)). \tag{2.4}$$

It consists of: root orientation $\boldsymbol{h}_t'^q \in \mathbb{R}^4$ in agent-centric coordinates; simulated pose $\boldsymbol{q}_t' \in \mathbb{R}^{74}$ ($\boldsymbol{q}_t' \triangleq (\boldsymbol{r}_t'^z, \boldsymbol{r}_t'^{rot}, \boldsymbol{j}_t^{\text{rot}})$, root height $\boldsymbol{r}_t'^z \in \mathbb{R}^1$, root orientation $\boldsymbol{r}_t'^{rot} \in \mathbb{R}^4$, and body pose $\boldsymbol{j}_t^{\text{rot}} \in \mathbb{R}^{69}$ expressed in Euler angles) in agent-centric coordinates; target pose $\widehat{\boldsymbol{q}}_t' \in \mathbb{R}^{74}$ in agent-centric coordinates; $(\boldsymbol{q}_t - \widehat{\boldsymbol{q}}_t) \in \mathbb{R}^{76}$ is the difference between the simulated and target pose (in world coordinate), calculated as $(\boldsymbol{q}_t - \widehat{\boldsymbol{q}}_t) \triangleq (\widehat{\boldsymbol{r}}_t^{\text{pos}} - \boldsymbol{r}_t^{\text{pos}}, \widehat{\boldsymbol{r}}_t^{\text{rot}} \ominus \boldsymbol{r}_t^{\text{rot}}, \widehat{\boldsymbol{j}}_t^{\text{rot}} - \boldsymbol{j}_t^{\text{rot}})$, where $\ominus$ calculates the rotation difference; $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{75}$ is the joint velocity computed by Mujoco; $(\psi_t - \widehat{\psi}_t) \in \mathbb{R}^1$ is the difference between the current heading (yaw) of the target and simulated root orientation; $\widehat{\boldsymbol{j}}_t'^{\text{pos}} \in \mathbb{R}^{72}$ and $(\boldsymbol{j}_t'^{\text{pos}} - \widehat{\boldsymbol{j}}_t'^{\text{pos}}) \in \mathbb{R}^{72}$ are joint position differences, calculated in the agent-centric space, respectively; $\widehat{\boldsymbol{j}}_t'^{\text{rot}} \in \mathbb{R}^{96}$ and $(\boldsymbol{j}_t'^{\text{rot}} \ominus \widehat{\boldsymbol{j}}_t'^{\text{rot}}) \in \mathbb{R}^{96}$ are joint rotation differences in quaternions (we first convert $\widehat{\boldsymbol{j}}_t'^{\text{rot}}$ from Euler angles to quaternions), calculated in the global and agent-centric space, respectively.

Reward function. The imitation reward function per timestep, similar to the reward defined in Yuan *et al*. [327] is as follows:

$$r_t = w_{\text{jr}} r_{\text{jr}} + w_{\text{jp}} r_{\text{jp}} + w_{\text{jv}} r_{\text{jv}} + w_{\text{res}} r_{\text{res}}, \tag{2.5}$$

where $w_{\text{jr}}, w_{\text{jp}}, w_{\text{jv}}, w_{\text{res}}$ are the weights of each reward. The joint rotation reward r_{jr} measures the difference between the simulated joint rotation $\boldsymbol{j}_t^{\text{rot}}$ and the target $\widehat{\boldsymbol{j}_t^{\text{rot}}}$ in quaternion for each joint on the humanoid. The joint position reward $\boldsymbol{r}_{\text{jp}}$ computes the distance between each joint's position $\boldsymbol{j}_t^{\text{pos}}$ and the target joint position $\widehat{\boldsymbol{j}_t^{\text{pos}}}$. The joint velocity reward $\boldsymbol{r}_{\text{jv}}$ penalizes the deviation of the estimated joint angular velocity $\dot{\boldsymbol{j}}_t^{\text{rot}}$ from the target $\hat{\dot{\boldsymbol{j}}}_t^{\text{rot}}$. The target velocity is computed from the data via finite difference. All above rewards include every joint on the humanoid model (including the root joint), and are calculated in the world coordinate frame. Finally, the residual force reward $\boldsymbol{r}_{\text{res}}$ encourages the policy to rely less on the external force and penalize for a large $\boldsymbol{\eta}_t$:

$$r_{\text{jr}} = \exp\left[-2.0\left(\|\boldsymbol{j}^{\text{rot}}{}_t \ominus \widehat{\boldsymbol{j}}_t^{\text{rot}}\|^2\right)\right], \quad r_{\text{jp}} = \exp\left[-5\left(\left\|\boldsymbol{j}_t^{\text{pos}} - \widehat{\boldsymbol{j}}_t^{\text{pos}}\right\|^2\right)\right],$$
$$r_{\text{jv}} = \exp[-0.005\left\|\dot{\boldsymbol{j}}_t^{\text{rot}} \ominus \hat{\dot{\boldsymbol{j}}}_t^{\text{rot}}\right\|^2], \quad r_{\text{res}} = \exp\left[-\left(\|\boldsymbol{\eta}_t\|^2\right)\right]. \tag{2.6}$$

We train our Universal Humanoid Controller for 10000 epoches, which takes about 5 days. Additional hyperparameters for training the UHC can be found in Table 2.6:

	γ	Batch Size	Value Learning Rate	Policy Learning Rate	PPO clip ϵ	Covariance Std
Value	0.95	50000	3×10^{-4}	5×10^{-5}	0.2	0.1
	w_{jr}	w_{jp}	w_{jv}	w_{res}	Sampling Temperature	
Value	0.3	0.55	0.1	0.05	2	

Table 2.6. Hyperparameters used for training the Universal Humanoid Controller.

Training data clearning We use the AMASS [176] dataset for training our UHC. The original AMASS dataset contains 13944 high-quality motion sequences, and around 2600 of them contain human-object interactions such as sitting on a chair, walking on a treadmill, and walking on a bench. Since AMASS does not contain object information, we can not faithfully recreate and simulate the human-object interactions. Thus, we use a combination of heuristics and visual inspection to remove these sequences. For instance, we detect sitting sequences through finding combinations of the humanoid's root, leg, and torso angles that correspond to the sitting posture; we find walking-on-a-bench sequences through detecting a prolonged airborne period; for sequences that are difficult to detect automatically, we conduct manual visual inspection. After the data cleaning process, we obtain 11299 motion sequences that do not contain human-object interaction for our UHC to learn from.

2.5.10 Evaluation on AMASS

To evaluate our Universal Humanoid Controller's ability to learn to imitate diverse human motion, we run our controller on the full AMASS dataset (after removing sequences that include human-object interactions) that we trained on. After data cleaning, the AMASS dataset contains 11299 high quality motion sequences, and contains challenging sequences such as kickboxing, dancing, backflipping, crawling, etc. We use a subset of metrics from egocentric pose estimation to evaluate the motion imitation results of UHC. Namely, we report S_{inter}, E_{root}, E_{mpjpe}, E_{acc}, where the human-object interation S_{inter} indicates whether the humanoid has become unstable and falls down during the imitation process. The baseline we compare against is the popular motion imitation method DeepMimic [208]. Since our framework uses a different physics simulation (Bullet [54] vs Mujoco [279]), we use an in-house implementation of DeepMimic. From the result of Table 2.7 we can see that our controller can imitate a large collection (10956/11299, 96.964%) of realistic human motion with high fidelity without falling. Our UHC also achieves very low joint position error on motion imitation and, upon visual inspection, our controller can imitate highly dynamic motion sequences such as dancing and kickboxing. Failure cases include some of the more challenging sequences such as breakdancing and cartwheeling and can be found in the supplementary video.

AMASS dataset				
Method	S_{inter} ↑	E_{root} ↓	E_{mpjpe} ↓	E_{acc} ↓
DeepMimic	24.0%	0.385	61.634	17.938
UHC w/o MCP	95.0%	0.134	25.254	5.383
UHC	**97.0%**	**0.133**	**24.454**	**4.460**

Table 2.7. Evaluation of motion imitation for our UHC using target motion from the AMASS dataset.

2.5.11 Evaluation on H36M

To evaluate our Universal Humanoid Controller's ability to generalize to *unseen motion sequences*, we use the popular Human 3.6M (H36M) dataset [114]. We first fit the SMPL body model to ground truth 3D keypoints similar to the process in [188] and obtain motion sequences in SMPL parameters. Notice that this fitting process is imperfect and the resulting motion sequence is of less quality than original MoCap sequences.

These sequences are also never seen by our UHC during training. As observed in Moon *et al*. [188], the fitted SMPL poses have a mean per joint position error of around 10mm. We use the train split of H36M (150 unique motion sequences) as the target pose for our UHC to mimic. From the results shown in Table 2.8, we can see that our UHC can imitate the unseen motion in H36M with high accuracy and success rate, and outperforms the baseline method significantly. Upon visual inspection, we can see that the failure cases often result from losing balance while the humanoid is crouching down or starts running suddenly. Since our controller does not use any sequence level information, it has no way of knowing the upcoming speedup of the target motion and can result in instability. This indicates the importance of the kinematic policy adjusting its target pose based on the current simulation state to prevent the humanoid from falling down, and signifies that further investigation is needed to obtain a better controller. For visual inspection of motion imitation quality and failure cases, please refer to our supplementary video.

H36M dataset				
Method	S_{inter} ↑	E_{root} ↓	E_{mpjpe} ↓	E_{acc} ↓
DeepMimic	0.0%	0.609	107.895	28.881
UHC w/o MCP	89.3%	0.200	**36.972**	4.723
UHC	**92.0%**	**0.194**	40.424	**3.672**

Table 2.8. Evaluation of motion imitation for our UHC using target motion from the H36M dataset.

2.5.12 Additional Dataset Details

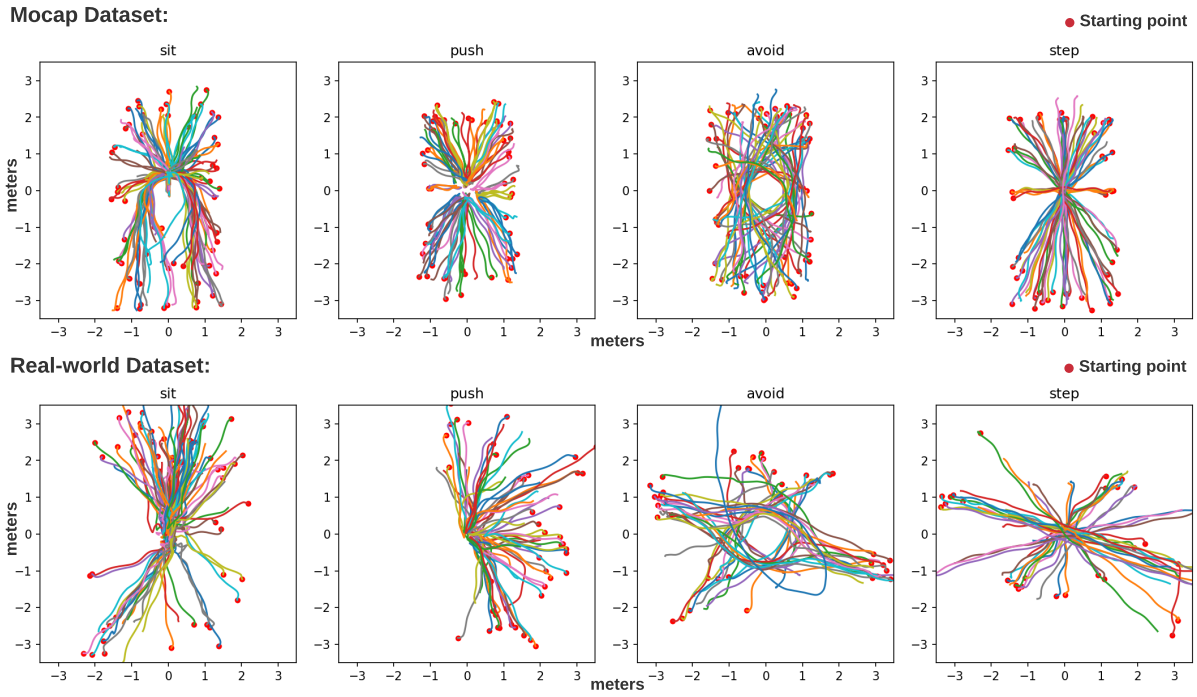


Figure 2.5. Trajectory analysis on our MoCap and real-world datasets. **Here we recenter each trajectory using the object position and plot the camera trajectory**, the objects are positioned differently across trajectories. The starting point is marked as a red dot.

2.5.13 MoCap dataset.

Our MoCap dataset (202 training sequences, 64 testing sequences, in total 148k frames) is captured in a MoCap studio with three different subjects. Each motion clip contains paired first-person footage of a person performing one of the five tasks: sitting down and (standing up from) a chair, avoiding an obstacle, stepping on a box, pushing a box, and generic locomotion (walking, running, crouching). Each action has around 50 sequences. The locomotion part of our dataset is merged from the egocentric dataset from EgoPose [324] since the two datasets are captured using a compatible system. MoCap markers are attached to the camera wearer and the objects to get the 3D full-body human pose and 6DoF object pose. To diversify the way actions are performed, we instruct the actors to vary their performance for each action (varying starting position and facing gait, speed *etc.*). We followed the Institutional Review Board's guidelines and obtained approval for the collection of this dataset. To study the diversity of our MoCap dataset, we plot the trajectory taken by the actors in Fig. 2.5. We can see that our trajectories are diverse and are spread out around a circle with varying distance from the objects. Table 2.9 shows the speed statistics for our MoCap dataset.

2.5.14 Real-world dataset.

Our real world dataset (183 testing sequences, in total 55k frames) is captured in everyday settings (living room and hallway) with an additional subject. It contains the same four types of interactions as our MoCap dataset and is captured from a head-mounted iPhone using a VR headset (demonstrated in Fig.2.6). Each action has around 40 sequences. As can be seen in the camera trajectory in Fig. 2.5, the real-world dataset is more heterogeneous than the MoCap dataset, and has more curves and banks overall. Speed analysis in Table 2.9 also shows that our real-world dataset has a larger standard deviation in terms of walking velocity and has a larger overall spread than the MoCap dataset. In all, our real-world dataset has more diverse trajectories and motion patterns than our MoCap dataset, and our dynamics-regulated kinematic policy can still estimate the sequences recorded in this dataset. To make sure that our off-the-shelf object detector and pose estimator can correctly register the object-of-interest, we ask the subject to look at the object at the beginning of each capture session as a calibration period. Later, we remove this calibration period and directly use the detected object pose. Notice that our framework is starting *position and orientation invariant*, since all of our input features are transformed into the agent-centric coordinate system using the transformation function $\boldsymbol{T}_{\text{AC}}$.

Figure 2.6. Our real-world dataset capturing equipment.

MoCap dataset					Real-world dataset				
Action	Mean	Min	Max	Std	Action	Mean	Min	Max	Std
Sit	0.646	0.442	0.837	0.098	Sit	0.556	0.227	0.891	0.171
Push	0.576	0.320	0.823	0.119	Push	0.526	0.234	0.762	0.127
Avoid	0.851	0.567	1.084	0.139	Avoid	0.668	0.283	0.994	0.219
Step	0.844	0.576	1.029	0.118	Step	0.729	0.395	1.092	0.196

Table 2.9. Speed analysis of our MoCap dataset and real-world dataset. Unit: (meters/second)

2.5.15 Dataset Diversity

2.6 Discussions

2.6.1 Failure Cases and Limitations

Although our method can produce realistic human pose and human-object interaction estimation from egocentric videos, we are still at the early stage of this challenging task. Our method performs well in the MoCap studio setting and generalizes to real-world settings, but is limited to a *predefined set* of interactions where we have data to learn from. Object class and pose information is computed by off-the-shelf methods such as Apple's ARkit [111], and is provided as a strong prior to our kinematic policy to infer pose. We also only factor in the 6DoF object pose in our state representation and discard all other object geometric information. The lower success rate on the real-world dataset also indicates that our method still suffers from covariate shifts and can become unstable when the shift becomes too extreme. Our Universal Humanoid Controller can imitate everyday motion with high accuracy, but can still fail at extreme motion. Due to the challenging nature of this task, in this work, we focus on developing a general framework to ground pose estimation with physics by merging the kinematics and dynamics aspects of human motion. To enable pose and human-object interaction estimation for arbitrary actions and objects, better scene understanding and kinematic motion planning techniques need to be developed.

2.6.2 Conclusion and Future Work

In this paper, we tackle, for the first time, estimating physically-plausible 3D poses from an egocentric video while the person is interacting with objects. We collect a motion capture dataset and real-world dataset to develop and evaluate our method, and extensive experiments have shown that our method outperforms all prior arts. We design a dynamics-regulated kinematic policy that can be directly trained and deployed inside a physics simulation, and we purpose a general-purpose humanoid controller that can be used in physics-based vision tasks easily. Through our real-world experiments, we show that it is possible to estimate 3D human poses and human-object interactions from just an egocentric view captured by consumer hardware (iPhone). In the future, we would like to support more action classes and further improve the robustness of our method by techniques such as using a learned motion prior. Applying

our dynamics-regulated training procedure to other vision tasks such as visual navigation and third-person pose estimation can also be of interest.

Chapter 3

PHC: Perpetual Humanoid Control for Real-time Simulated Avatars

3.1 Introduction

While UHC can imitate large-scale human motion, the dependency on using residual force compromises physical realism. In this chapter, we aim to relieve this dependency. The first major challenge is imitating a large-scale motion dataset with a high success rate. While reinforcement learning (RL)-based imitation policies have shown promising results, successfully imitating motion from a large dataset, such as AMASS (ten thousand clips, 40 hours of motion), with a *single* policy has yet to be achieved. Attempts to use larger or a mixture of expert policies have been met with some success [292, 296], although they have not yet scaled to the largest dataset. Therefore, researchers have resorted to using external forces to help stabilize the humanoid. Residual force control (RFC) [327] has helped to create motion imitators that can mimic up to 97% of the AMASS dataset [172], and has seen successful applications in human pose estimation from video [84, 173, 329] and language-based motion generation [328]. However, the external force compromises physical realism by acting as a "hand of God" that puppets the humanoid, leading to artifacts such as flying and floating. One might argue that, with RFC, the realism of simulation is compromised, as the model can freely apply a non-physical force on the humanoid.

Another important aspect of motion imtiation is how to handle noisy input and failure cases. In this work, we consider human poses estimated from video or language input. Especially with respect to video input, artifacts such as floating [328], foot sliding [349], and physically impossible poses are prevalent in popular pose estimation methods due to occlusion, challenging view point and lighting, fast motions *etc*. To handle these cases, most physics-based methods resort to resetting the humanoid when a failure condition is triggered [172, 175, 325]. However, resetting successfully requires a high-quality reference pose, which is often difficult to obtain due to the noisy nature of the pose estimates, leading to a vicious cycle of falling and resetting to unreliable poses. Thus, it is important to have a controller that can gracefully handle unexpected falls and noisy input, naturally recover from fail-state, and resume imitation.

In this work, our aim is to create a humanoid controller specifically designed to control

real-time virtual avatars, where video observations of a human user are used to control the avatar. We design the Perpetual Humanoid Controller (PHC), a *single* policy that achieves a high success rate on motion imitation **and** can recover from fail-state naturally. We propose a progressive multiplicative control policy (PMCP) to learn from motion sequences in the entire AMASS dataset without suffering catastrophic forgetting. By treating harder and harder motion sequences as a different "task" and gradually allocating new network capacity to learn, PMCP retains its ability to imitate easier motion clips when learning harder ones. PMCP also allows the controller to learn fail-state recovery tasks *without compromising* its motion imitation capabilities. Additionally, we adopt Adversarial Motion Prior (AMP) [215] throughout our pipeline and ensure natural and human-like behavior during fail-state recovery. Furthermore, while most motion imitation methods require both estimates of link position and rotation as input, we design controllers that require only the link positions. This input can be generated more easily by vision-based 3D keypoint estimators or 3D pose estimates from VR controllers.

To summarize, our contributions are as follows: (1) we propose a Perpetual Humanoid Controller that can successfully imitate 98.9% of the AMASS dataset without applying any external forces; (2) we propose the progressive multiplicative control policy to learn from a large motion dataset without catastrophic forgetting and unlock additional capabilities such as fail-state recovery; (3) our controller is task-agnostic and is compatible with off-the-shelf video-based pose estimators as a drop-in solution. We demonstrate the capabilities of our controller by evaluating on both Motion Capture (MoCap) and estimated motion from videos. We also show a live (30 fps) demo of driving perpetually simulated avatars using a webcam video as input.

3.2 Related Works

Physics-based Motion Imitation. Governed by the laws of physics, simulated characters [19, 46, 79, 84, 93, 184, 208, 211–213, 215, 292, 295, 327] have the distinct advantage of creating natural human motion, human-to-human interaction [158, 298], and human-object interactions [184, 213]. Since most modern physics simulators are not differentiable, training these simulated agents requires RL, which is time-consuming & costly. As a result, most of the work focuses on small-scale use cases such as interactive control based on user input [19, 213, 215, 292], playing sports [158, 184, 298], or other modular tasks (reaching goals [299], dribbling [215], moving around [211], *etc.*). On the other hand, imitating large-scale motion datasets is a challenging yet fundamental task, as an agent that can imitate reference motion can be easily paired with a motion generator to achieve different tasks. From learning to imitate a single clip [208] to datasets [46, 285, 292, 297], motion imitators have demonstrated their impressive ability to imitate reference motion, but are often limited to imitating high-quality MoCap data. Among them, ScaDiver [297] uses a mixture of expert policy to scale up to the CMU MoCap dataset and achieves a success rate of around 80% measured by time to failure. Unicon [292] shows qualitative results in imitation and transfer, but does not quantify the imitator's ability to imitate clips from datasets. MoCapAct [285] first learns single-clip experts on the CMU MoCap dataset, and distills them into a single that achieves around 80% of the experts' performance. The effort closest to ours is UHC [172], which successfully imitates 97% of the AMASS dataset.

However, UHC uses residual force control [325], which applies a non-physical force at the root of the humanoid to help balance. Although effective in preventing the humanoid from falling, RFC reduces physical realism and creates artifacts such as floating and swinging, especially when motion sequences become challenging [172,173]. Compared to UHC, our controller does not utilize any external force.

Fail-state Recovery for Simulated Characters. As simulated characters can easily fall when losing balance, many approaches [46, 213, 251, 270, 325] have been proposed to help recovery. PhysCap [251] uses a floating-base humanoid that does not require balancing. This compromises physical realism, as the humanoid is no longer properly simulated. Egopose [325] designs a fail-safe mechanism to reset the humanoid to the kinematic pose when it is about to fall, leading to potential teleport behavior in which the humanoid keeps resetting to unreliable kinematic poses. NeruoMoCon [108] utilizes sampling-based control and reruns the sampling process if the humanoid falls. Although effective, this approach does not guarantee success and prohibits real-time use cases. Another natural approach is to use an additional recovery policy [46] when the humanoid has deviated from the reference motion. However, since such a recovery policy no longer has access to the reference motion, it produces unnatural behavior, such as high-frequency jitters. To combat this, ASE [213] demonstrates the ability to rise naturally from the ground for a sword-swinging policy. While impressive, in motion imitation the policy not only needs to get up from the ground, but also goes back to tracking the reference motion. In this work, we propose a comprehensive solution to the fail-state recovery problem in motion imitation: our PHC can rise from fallen state and naturally walks back to the reference motion and resume imitation.

Progressive Reinforcement Learning. When learning from data containing diverse patterns, catastrophic forgetting [74, 179] is observed when attempting to perform multi-task or transfer learning by fine-tuning. Various approaches [62, 118, 131] have been proposed to combat this phenomenon, such as regularizing the weights of the network [131], learning multiple experts [118], or increasing the capacity using a mixture of experts [248, 296, 345] or multiplicative control [212]. A paradigm has been studied in transfer learning and domain adaption as progressive learning [33, 39] or curriculum learning [17]. Recently, progressive reinforcement learning [20] has been proposed to distill skills from multiple expert policies. It aims to find a policy that best matches the action distribution of experts instead of finding an optimal mix of experts. Progressive Neural Networks (PNN) [242] proposes to avoid catastrophic forgetting by freezing the weights of the previously learned subnetworks and initializing additional subnetworks to learn new tasks. The experiences from previous subnetworks are forwarded through lateral connections. PNN requires manually choosing which subnetwork to use based on the task, preventing it from being used in motion imitation since reference motion does not have the concept of task labels.

3.3 Method

In Sec.3.3.1, we first set up the preliminary of our main framework. Sec.3.3.2 describes our progressive multiplicative control policy to learn to imitate a large dataset of human motion

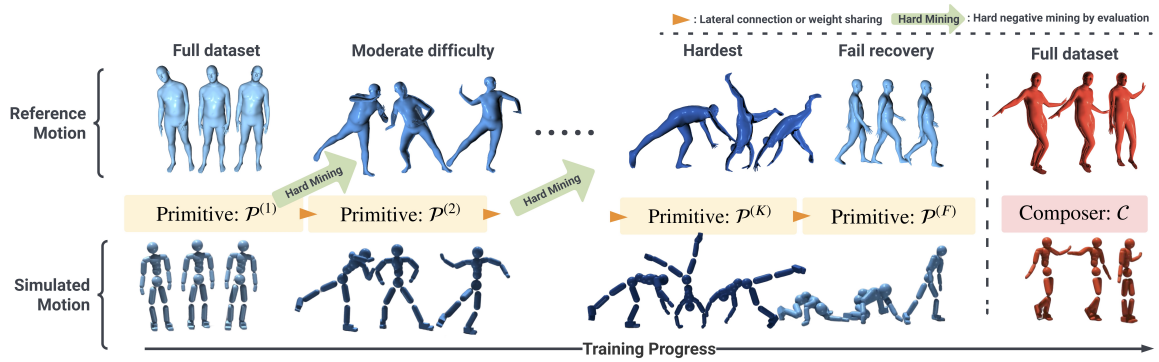


Figure 3.1. Our progressive training procedure to train primitives $\mathcal{P}^{(1)}, \mathcal{P}^{(2)}, \cdots, \mathcal{P}^{(K)}$ by gradually learning harder and harder sequences. Fail recovery $\mathcal{P}^{(F)}$ is trained in the end on simple locomotion data; a composer is then trained to combine these frozen primitives.

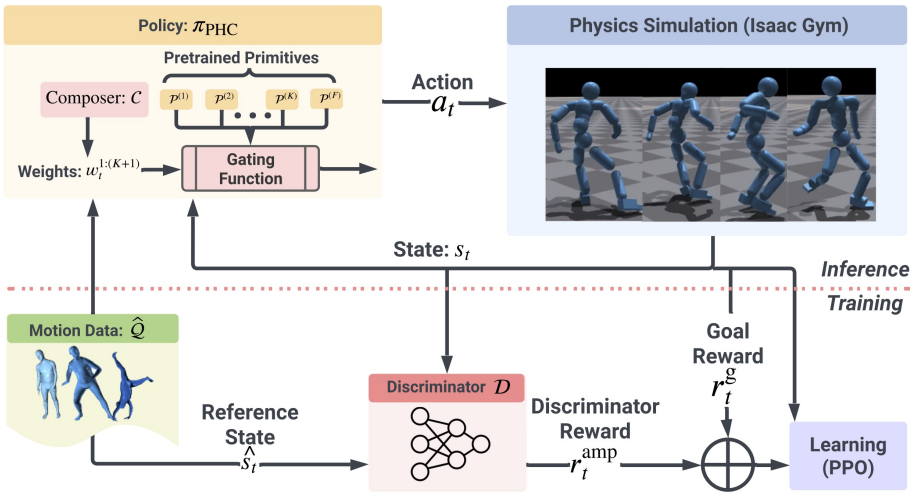


Figure 3.2. Goal-conditioned RL framework with Adversarial Motion Prior. Each primitive $\mathcal{P}^{(k)}$ and composer $\mathcal{C}$ is trained using the same procedure, and here we visualize the final product $\pi_{\text{Omnigrasp}}$.

and recover from fail-states. Finally, in Sec.3.3.3, we briefly describe how we connect our task-agnostic controller to off-the-shelf video pose estimators and generators for real-time use cases.

3.3.1 Goal Conditioned Motion Imitation with Adversarial Motion Prior

Our controller follows the general framework of goal-conditioned RL (Fig.3.2), where a goal-conditioned policy π_{PHC} is tasked to imitate reference motion $\hat{\boldsymbol{q}}_{\boldsymbol{1:t}}$ or keypoints $\hat{\boldsymbol{p}}_{1:T}$. Similar to prior work [172, 208], we formulate the task as a Markov Decision Process (MDP).

State. The simulation state $\boldsymbol{s}_t \triangleq (\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ consists of humanoid proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$. Proprioception $\boldsymbol{s}_t^{\text{p}} \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{\beta})$ contains the 3D body pose $\boldsymbol{q}_t$, velocity $\dot{\boldsymbol{q}}_t$, and (optionally) body shapes $\boldsymbol{\beta}$. When trained with different body shapes, $\boldsymbol{\beta}$ contains information about the length of the limb of each body link [175]. For rotation-based motion imitation, the goal state $\boldsymbol{s}_t^{\text{g}}$ is defined as the difference between the next time step reference quantitives and their

simulated counterpart:

$$\boldsymbol{s}_t^{\text{g-rot}} \triangleq (\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_{t+1} - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_{t+1}, \hat{\boldsymbol{p}}_{t+1})$$

where $\ominus$ calculates the rotation difference. For keypoint-only imtiation, the goal state becomes

$$\boldsymbol{s}_t^{\text{g-kp}} \triangleq (\hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{p}}_{t+1}).$$

All of the above quantities in $\boldsymbol{s}_t^{\text{g}}$ and $\boldsymbol{s}_t^{\text{p}}$ are normalized with respect to the humanoid's current facing direction and root position [172, 299].

Reward. Unlike prior motion tracking policies that only use a motion imitation reward, we use the recently proposed Adversarial Motion Prior [215] and include a discriminator reward term throughout our framework. Including the discriminator term helps our controller produce stable and natural motion and is especially crucial in learning natural fail-state recovery behaviors. Specifically, our reward is defined as the sum of a task reward r_t^{g}, a style reward r_t^{amp}, and an additional energy penalty r_t^{energy} [208]:

$$r_t = 0.5 r_t^{\text{g}} + 0.5 r_t^{\text{amp}} + r_t^{\text{energy}}. \tag{3.1}$$

For the discriminator, we use the same observations, loss formulation, and gradient penalty as AMP [215]. The energy penalty is expressed as $-0.0005 \cdot \sum_{j \in \text{ joints}} |\boldsymbol{\mu_j} \boldsymbol{\omega_j}|^2$ where $\boldsymbol{\mu_j}$ and $\boldsymbol{\omega_j}$ correspond to the joint torque and the joint angular velocity, respectively. The energy penalty [76] regulates the policy and prevents high-frequency jitter of the foot that can manifest in a policy trained without external force (see Sec.3.4.1). The task reward is defined based on the current training objective, which can be chosen by switching the reward function for motion imitation $\boldsymbol{\mathcal{R}}^{\text{imitation}}$ and fail-state recovery $\boldsymbol{\mathcal{R}}^{\text{recover}}$. For motion tracking, we use:

$$r_t^{\text{g-imitation}} = \boldsymbol{\mathcal{R}}^{\text{imitation}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_t) = w_{\text{jp}} e^{-100\|\hat{\boldsymbol{p}}_t - \boldsymbol{p}_t\|} + w_{\text{jr}} e^{-10\|\hat{\boldsymbol{q}}_t \ominus \boldsymbol{q}_t\|} + w_{\text{jv}} e^{-0.1\|\hat{\boldsymbol{v}}_t - \boldsymbol{v}_t\|} + w_{\text{j}\omega} e^{-0.1\|\hat{\boldsymbol{\omega}}_t - \boldsymbol{\omega}_t\|} \tag{3.2}$$

where we measure the difference between the translation, rotation, linear velocity, and angular velocity of the rigid body for all links in the humanoid. For fail-state recovery, we define the reward $r_t^{\text{g-recover}}$ in Eq.3.3.

Action. We use a proportional derivative (PD) controller at each DoF of the humanoid and the action $\boldsymbol{a}_t$ specifies the PD target. With the target joint set as $\boldsymbol{q}_t^d = \boldsymbol{a}_t$, the torque applied at each joint is $\boldsymbol{\tau}^i = \boldsymbol{k}^p \circ (\boldsymbol{a}_t - \boldsymbol{q}_t) - \boldsymbol{k}^d \circ \dot{\boldsymbol{q}}_t$. Notice that this is different from the residual action representation [172, 202, 327] used in prior motion imitation methods, where the action is added to the reference pose: $\boldsymbol{q}_t^d = \hat{\boldsymbol{q}}_t + \boldsymbol{a}_t$ to speed up training. As our PHC needs to remain robust to noisy and ill-posed reference motion, we remove such a dependency on reference motion in our action space. We do not use any external forces [327] or meta-PD control [329].

Control Policy and Discriminator. Our control policy $\pi_{\text{PHC}}(\boldsymbol{a}_t | \boldsymbol{s}_t) = \mathcal{N}(\mu(\boldsymbol{s}_t), \sigma)$ represents a Gaussian distribution with fixed diagonal covariance. The AMP discriminator $\boldsymbol{\mathcal{D}}(\boldsymbol{s}_{t-10:t}^{\text{p}})$ computes a real and fake value based on the current prioproception of the humanoid. All of our networks (discriminator, primitive, value function, and discriminator) are two-layer multilayer perceptrons (MLP) with dimensions [1024, 512].

Humanoid. Our humanoid controller can support any human kinematic structure, and we use the SMPL [161] kinematic structure following prior arts [172,173,329]. The SMPL body contains 24 rigid bodies, of which 23 are actuated, resulting in an action space of $\boldsymbol{a}_t \in \mathbb{R}^{23\times 3}$. The body proportion can vary based on a body shape parameter $\beta \in \mathbb{R}^{10}$.

Initialization and Relaxed Early Termination. We use reference state initialization (RSI) [208] during training and randomly select a starting point for a motion clip for imitation. For early termination, we follow UHC [172] and terminate the episode when the joints are more than 0.5 meters globally on average from the reference motion. Unlike UHC, we remove the ankle and toe joints from the termination condition. As observed by RFC [327], there exists a dynamics mismatch between simulated humanoids and real humans, especially since the real human foot is multisegment [201]. Thus, it is not possible for the simulated humanoid to have the exact same foot movement as MoCap, and blindly following the reference foot movement may lead to the humanoid losing balance. Thus, we propose Relaxed Early Termination (RET), which allows the humanoid's ankle and toes to slightly deviate from the MoCap motion to remain balanced. Notice that the humanoid still receives imitation and discriminator rewards for these body parts, which prevents these joints from moving in a nonhuman manner. We show that though this is a small detail, it is conducive to achieving a good motion imitation success rate.

Hard Negative Mining. When learning from a large motion dataset, it is essential to train on harder sequences in the later stages of training to gather more informative experiences. We use a similar hard negative mining procedure as in UHC [172] and define hard sequences by whether or not our controller can successfully imitate this sequence. From a motion dataset $\hat{\boldsymbol{Q}}$, we find hard sequences $\hat{\boldsymbol{Q}}_{\text{hard}} \subseteq \hat{\boldsymbol{Q}}$ by evaluating our model over the entire dataset and choosing sequences that our policy fails to imitate.

3.3.2 Progressive Multiplicative Control Policy

As training continues, we notice that the performance of the model plateaus as it forgets older sequences when learning new ones. Hard negative mining alleviates the problem to a certain extent, yet suffers from the same issue. Introducing new tasks, such as fail-state recovery, may further degrade imitation performance due to catastrophic forgetting. Thus, we propose a progressive multiplicative control policy (PMCP), which allocates new subnetworks (primitives $\boldsymbol{\mathcal{P}}$) to learn harder sequences.

Progressive Neural Networks (PNN). A PNN [242] starts with a single primitive network $\boldsymbol{\mathcal{P}}^{(1)}$ trained on the full dataset $\hat{\boldsymbol{Q}}$. Once $\boldsymbol{\mathcal{P}}^{(1)}$ is trained to convergence on the entire motion dataset $\hat{\boldsymbol{Q}}$ using the imitation task, we create a subset of hard motions by evaluating $\boldsymbol{\mathcal{P}}^{(1)}$ on $\hat{\boldsymbol{Q}}$. We define convergence as the success rate on $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}}$ no longer increases. The sequences that $\boldsymbol{\mathcal{P}}^{(1)}$ fails on is formed as $\hat{\boldsymbol{Q}}^{(1)}_{\text{hard}}$. We then freeze the parameters of $\boldsymbol{\mathcal{P}}^{(1)}$ and create a new primitive $\boldsymbol{\mathcal{P}}^{(2)}$ (randomly initialized) along with lateral connections that connect each layer of $\boldsymbol{\mathcal{P}}^{(1)}$ to $\boldsymbol{\mathcal{P}}^{(2)}$. During training, we construct each $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}}$ by selecting the failed sequences from the previous step $\hat{\boldsymbol{Q}}^{(k-1)}_{\text{hard}}$, resulting in a smaller and smaller hard subset: $\hat{\boldsymbol{Q}}^{(k)}_{\text{hard}} \subseteq \hat{\boldsymbol{Q}}^{(k-1)}_{\text{hard}}$. In this way, we ensure that each newly initiated primitive $\boldsymbol{\mathcal{P}}^{(k)}$ is responsible for learning a new and harder subset of motion sequences, as can be seen in Fig.3.1. Notice that this is different from

Algo 3: Learn Progressive Multiplicative Control Policy

Function `TrainPPO`(π, $\hat{\boldsymbol{Q}}^{(k)}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{R}}$)**:**
  while *not converged* **do**
    $\boldsymbol{M} \leftarrow \emptyset$ initialize sampling memory ;
    while $\boldsymbol{M}$ *not full* **do**
      $\hat{\boldsymbol{q}}_{1:T} \leftarrow$ sample motion from $\hat{\boldsymbol{Q}}$;
      for $t \leftarrow 1...T$ **do**
        $\boldsymbol{s}_t \leftarrow \left(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{g}\right)$;
        $\boldsymbol{a}_t \sim \pi(\boldsymbol{a}_t|\boldsymbol{s}_t)$;
        $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulation`;
        $\boldsymbol{r}_t \leftarrow \boldsymbol{\mathcal{R}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_{t+1})$;
        store $(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$ into memory $\boldsymbol{M}$;
    $\boldsymbol{\mathcal{P}}^{(k)}, \boldsymbol{\mathcal{V}} \leftarrow$ PPO update using experiences collected in $\boldsymbol{M}$;
    $\boldsymbol{\mathcal{D}} \leftarrow$ Discriminator update using experiences collected in $\boldsymbol{M}$
  return π ;

Input: Ground truth motion dataset $\hat{\boldsymbol{Q}}$;
$\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{V}}, \hat{\boldsymbol{Q}}_{\text{hard}}^{(1)} \leftarrow \hat{\boldsymbol{Q}}$ `// Initialize discriminator, value function, and dataset`;
for $k \leftarrow 1...K$ **do**
  Initialize $\boldsymbol{\mathcal{P}}^{(k)}$ `// Lateral connection/weight sharing`;
  $\boldsymbol{\mathcal{P}}^{(k)} \leftarrow$ `TrainPPO`($\boldsymbol{\mathcal{P}}^{(k)}$, $\hat{\boldsymbol{Q}}_{hard}^{(k)}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{R}}^{imitation}$) ;
  $\hat{\boldsymbol{Q}}_{\text{hard}}^{(k+1)} \leftarrow$ eval($\boldsymbol{\mathcal{P}}^{(k)}$, $\hat{\boldsymbol{Q}}^{(k)}$);
  $\boldsymbol{\mathcal{P}}^{(k)} \leftarrow$ freeze $\boldsymbol{\mathcal{P}}^{(k)}$;
$\boldsymbol{\mathcal{P}}^{(F)} \leftarrow$ `TrainPPO`($\boldsymbol{\mathcal{P}}^{(F)}$, $\boldsymbol{Q}^{loco}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{R}}^{recover}$) `// Fail-state Recovery`;
$\pi_{\text{Omnigrasp}} \leftarrow \{\boldsymbol{\mathcal{P}}^{(1)} \cdots \boldsymbol{\mathcal{P}}^{(K)}, \boldsymbol{\mathcal{P}}^{(F)}, \boldsymbol{\mathcal{C}}\}$;
$\pi_{\text{Omnigrasp}} \leftarrow$ `TrainPPO`($\pi_{Omnigrasp}$, $\hat{\boldsymbol{Q}}$, $\boldsymbol{\mathcal{D}}$, $\boldsymbol{\mathcal{V}}$, $\{\boldsymbol{\mathcal{R}}^{imitation}, \boldsymbol{\mathcal{R}}^{recover}\}$) `// Train Composer`;

hard-negative mining in UHC [172], as we initialize a new primitive $\boldsymbol{\mathcal{P}}^{(k+1)}$ to train. Since the original PNN is proposed to solve completely new tasks (such as different Atari games), a lateral connection mechanism is proposed to allow later tasks to choose between reuse, modify, or discard prior experiences. However, mimicking human motion is highly correlated, where fitting to harder sequences $\hat{\boldsymbol{Q}}_{\text{hard}}^{(k)}$ can effectively draw experiences from previous motor control experiences. Thus, we also consider a variant of PNN where there are **no lateral** connections, but the new primitives are initialized from the weights of the prior layer. This weight sharing scheme is similar to fine-tuning on the harder motion sequences using a new primitive $\boldsymbol{\mathcal{P}}^{(k+1)}$ and preserve $\boldsymbol{\mathcal{P}}^{(k)}$'s ability to imitate learned sequences.

Fail-state Recovery. In addition to learning harder sequences, we also learn new tasks, such as recovering from fail-state. We define three types of fail-state: 1) fallen on the ground; 2) faraway from the reference motion ($> 0.5m$); 3) their combination: fallen and faraway. In these situations, the humanoid should get up from the ground, approach the reference motion in a natural way, and resume motion imitation. For this new task, we initialize a primitive $\boldsymbol{\mathcal{P}}^{(F)}$ at the end of the primitive stack. $\boldsymbol{\mathcal{P}}^{(F)}$ shares the same input and output space as $\boldsymbol{\mathcal{P}}^{(1)} \cdots \boldsymbol{\mathcal{P}}^{(k)}$, but since the reference motion does not provide useful information about fail-state recovery (the humanoid should not attempt to imitate the reference motion when lying on the ground), we modify the state space during fail-state recovery to remove all information about the reference

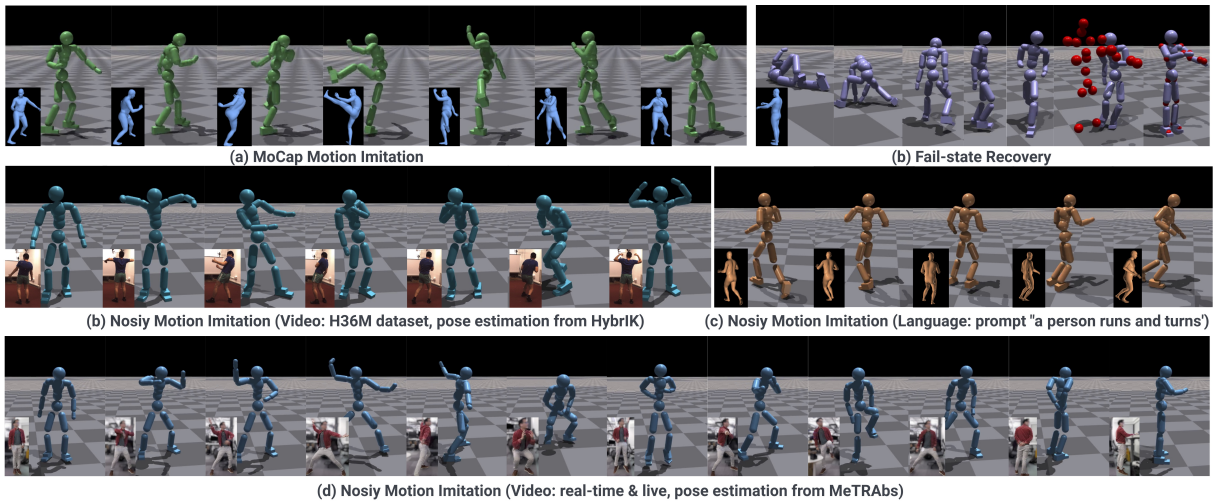


Figure 3.3. (a) Imitating high-quality MoCap – spin and kick. (b) Recover from fallen state and go back to reference motion (indicated by red dots). (b) Imitating noisy motion estimated from video. (c) Imitating motion generated from language. (d) Using poses estimated from a webcam stream for a real-time simulated avatar.

motion except the root. For the reference joint rotation $\hat{\boldsymbol{\theta}}_t = [\hat{\boldsymbol{\theta}}_t^0, \hat{\boldsymbol{\theta}}_t^1, \cdots \hat{\boldsymbol{\theta}}_t^J]$ where $\hat{\boldsymbol{\theta}}_t^i$ corresponds to the i^{th} joint, we construct $\hat{\boldsymbol{\theta}}'_t = [\hat{\boldsymbol{\theta}}_t^0, \boldsymbol{\theta}_t^1, \cdots \boldsymbol{\theta}_t^j]$ where all joint rotations except the root are replaced with simulated values (without $\hat{\cdot}$). This amounts to setting the non-root joint goals to be identity when computing the goal states: $\boldsymbol{s}_t^{\text{g-Fail}} \triangleq (\hat{\boldsymbol{\theta}}'_t \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t, \hat{\boldsymbol{v}}'_t - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}'_t - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}'_t, \hat{\boldsymbol{p}}'_t)$. $\boldsymbol{s}_t^{\text{g-Fail}}$ thus collapse from an imitation objective to a point-goal [299] objective where the only information provided is the relative position and orientation of the target root. When the reference root is too far ($> 5m$), we normalize $\hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t$ as $\frac{5\times(\hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t)}{\|\hat{\boldsymbol{p}}'_t - \boldsymbol{p}_t\|_2}$ to clamp the goal position. Once the humanoid is close enough ($e.g. < 0.5m$), the goal will switch back to full-motion imitation:

$$\boldsymbol{s}_t^{\text{g}} = \begin{cases} \boldsymbol{s}_t^{\text{g}} & \|\hat{\boldsymbol{p}}_t^0 - \boldsymbol{p}_t^0\|_2 \leq 0.5 \\ \boldsymbol{s}_t^{\text{g-Fail}} & \text{otherwise.} \end{cases} \tag{3.3}$$

To create fallen states, we follow ASE [213] and randomly drop the humanoid on the ground at the beginning of the episode. The faraway state can be created by initializing the humanoid $2 \sim 5$ meters from the reference motion. The reward for fail-state recovery consists of the AMP reward r_t^{amp}, point-goal reward $r_t^{\text{g-point}}$, and energy penalty r_t^{energy}, calculated by the reward function $\boldsymbol{\mathcal{R}}^{\text{recover}}$:

$$r_t^{\text{g-recover}} = \boldsymbol{\mathcal{R}}^{\text{recover}}(\boldsymbol{s}_t, \hat{\boldsymbol{q}}_t) = 0.5 r_t^{\text{g-point}} + 0.5 r_t^{\text{amp}} + 0.1 r_t^{\text{energy}}, \tag{3.4}$$

The point-goal reward is formulated as $r_t^{\text{g-point}} = (d_{t-1} - d_t)$ where d_t is the distance between the root reference and simulated root at the time step t [299]. For training $\boldsymbol{\mathcal{P}}^{(F)}$, we use a hand-picked subset of the AMASS dataset named $\boldsymbol{Q}^{\text{loco}}$ where it contains mainly walking and running sequences. Learning using only $\boldsymbol{Q}^{\text{loco}}$ coaxes the discriminator $\boldsymbol{\mathcal{D}}$ and the AMP reward r_t^{amp} to bias toward simple locomotion such as walking and running. We do not initialize a new value function and discriminator while training the primitives and continuously fine-tune the existing ones.

Multiplicative Control. Once each primitive has been learned, we obtain $\{\boldsymbol{\mathcal{P}}^{(1)} \cdots \boldsymbol{\mathcal{P}}^{(K)}, \boldsymbol{\mathcal{P}}^{(F)}\}$, with each primitive capable of imitating a subset of the dataset $\hat{\boldsymbol{Q}}$. In Progressive Networks [242], task switching is performed manually. In motion imitation, however, the boundary between hard and easy sequences is blurred. Thus, we utilize Multiplicative Control Policy (MCP) [212] and train an additional composer $\boldsymbol{\mathcal{C}}$ to dynamically combine the learned primitives. Essentially, we use the pretrained primitives as a informed search space for the composer $\boldsymbol{\mathcal{C}}$, and $\boldsymbol{\mathcal{C}}$ only needs to select which primitives to activate for imitation. Specifically, our composer $\boldsymbol{\mathcal{C}}(\boldsymbol{w}_t^{1:K+1}|\boldsymbol{s}_t)$ consumes the same input as the primitives and outputs a weight vector $\boldsymbol{w}_t^{1:K+1} \in \mathbb{R}^{k+1}$ to activate the primitives. Combining our composer and primitives, we have the PHC's output distribution:

$$\pi_{\text{PHC}}(\boldsymbol{a}_t \mid \boldsymbol{s}_t) = \frac{1}{\boldsymbol{\mathcal{C}}(\boldsymbol{s}_t)} \prod_i^k \boldsymbol{\mathcal{P}}^{(i)}(\boldsymbol{a}_t^{(i)} \mid \boldsymbol{s}_t)^{\boldsymbol{\mathcal{C}}(\boldsymbol{s}_t)}, \quad \boldsymbol{\mathcal{C}}(\boldsymbol{s}_t) \geq 0. \tag{3.5}$$

As each $\boldsymbol{\mathcal{P}}^{(k)}$ is an independent Gaussian, the action distribution:

$$\mathcal{N}\left(\frac{1}{\sum_l^k \frac{\boldsymbol{\mathcal{C}}_i(\boldsymbol{s}_t)}{\sigma_l^j(\boldsymbol{s}_t)}} \sum_i^k \frac{\boldsymbol{\mathcal{C}}_i(\boldsymbol{s}_t)}{\sigma_i^j(\boldsymbol{s}_t)} \mu_i^j(\boldsymbol{s}_t), \sigma^j(\boldsymbol{s}_t) = \left(\sum_i^k \frac{\boldsymbol{\mathcal{C}}_i(\boldsymbol{s}_t)}{\sigma_i^j(\boldsymbol{s}_t)}\right)^{-1}\right), \tag{3.6}$$

where $\mu_i^j(\boldsymbol{s}_t)$ corresponds to the $\boldsymbol{\mathcal{P}}^{(\text{i})}$'s j^{th} action dimension. Unlike a Mixture of Expert policies that only activates one at a time (top-1 MOE), MCP combines the actors' distribution and activates all actors at the same (similar to top-inf MOE). Unlike MCP, we progressively train our primitives and make the composer and actor share the same input space. Since primitives are independently trained for different harder sequences, we observe that the composite policy sees a significant boost in performance. During composer training, we interleave fail-state recovery training. The training process is described in Alg.3 and Fig.3.1.

3.3.3 Connecting with Motion Estimators

Our PHC is task-agnostic as it only requires the next time-step reference pose $\tilde{\boldsymbol{q}}_t$ or the keypoint $\tilde{\boldsymbol{p}}_t$ for motion tracking. Thus, we can use any off-the-shelf video-based human pose estimator or generator compatible with the SMPL kinematic structure. For driving simulated avatars from videos, we employ HybrIK [143] and MeTRAbs [262,263], both of which estimate in the metric space with the important distinction that HybrIK outputs joint rotation $\tilde{\boldsymbol{\theta}}_t$ while MeTRAbs only outputs 3D keypoints $\tilde{\boldsymbol{p}}_t$. For language-based motion generation, we use the Motion Diffusion Model (MDM) [277]. MDM generates disjoint motion sequences based on prompts, and we use our controller's recovery ability to achieve in-betweening.

3.4 Experiments

We evaluate and ablate our humanoid controller's ability to imitate high-quality MoCap sequences and noisy motion sequences estimated from videos in Sec.3.4.1. In Sec.3.4.2, we test

		AMASS-Train*					AMASS-Test*					H36M-Motion*				
Method	RFC	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
UHC	✓	97.0 %	36.4	25.1	4.4	5.9	96.4 %	50.0	31.2	9.7	12.1	87.0%	59.7	35.4	4.9	7.4
UHC	✗	84.5 %	62.7	39.6	10.9	10.9	62.6%	58.2	98.1	22.8	21.9	23.6%	133.14	67.4	14.9	17.2
Ours	✗	**98.9 %**	**37.5**	**26.9**	**3.3**	**4.9**	96.4%	**47.4**	**30.9**	6.8	**9.1**	92.9%	50.3	**33.3**	3.7	**5.5**
Ours-kp	✗	98.7%	40.7	32.3	3.5	5.5	**97.1%**	53.1	39.5	**7.5**	10.4	**95.7%**	**49.5**	39.2	**3.7**	5.8

Table 3.1. Quantitative results on imitating MoCap motion sequences (* indicates removing sequences containing human-object interaction). AMASS-Train*, AMASS-Test*, and H36M-Motion* contains 11313, 140, and 140 high-quality MoCap sequences, respectively.

			H36M-Test-Video*		
Method	RFC	Pose Estimate	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓
UHC	✓	HybrIK + MeTRAbs (root)	58.1%	75.5	49.3
UHC	✗	HybrIK + MeTRAbs (root)	18.1%	126.1	67.1
Ours	✗	HybrIK + MeTRAbs (root)	88.7%	**55.4**	**34.7**
Ours-kp	✗	HybrIK + MeTRAbs (root)	90.0%	55.8	41.0
Ours-kp	✗	MeTRAbs (all keypoints)	**91.9%**	55.7	41.1

Table 3.2. Motion imitation on noisy motion. We use HybrIK [143] to estimate the joint rotations $\tilde{\boldsymbol{\theta}}_t$ and uses MeTRAbs [263] for global 3D keypoints $\tilde{\boldsymbol{p}}_t$. HybrIK + MeTRAbs (root): using joint rotations $\tilde{\boldsymbol{\theta}}_t$ from HybrIK and root position $\tilde{\boldsymbol{p}}_t^0$ from MeTRAbs. MeTRAbs (all keypoints): using all keypoints $\tilde{\boldsymbol{p}}_t$ from MeTRAbs, only applicable to our keypoint-based controller.

our controller's ability to recovery from fail-state. As motion is best in videos, we provide extensive qualitative results on the website. All experiments are run three times and averaged.

Baselines. We compare with the SOTA motion imitator UHC [172] and use the official implementation. We compare against UHC both *with and without* residual force control.

Implementation Details. We uses four primitives (including fail-state recovery) for all our evaluations. PHC can be trained on a single NVIDIA A100 GPU; it takes around a week to train all primitives and the composer. Once trained, the composite policy runs at > 30 FPS. Physics simulation is carried out in NVIDIA's Isaac Gym [177]. The control policy is run at 30 Hz, while simulation runs at 60 Hz. For evaluation, we do not consider body shape variation and use the mean SMPL body shape.

Datasets. PHC is trained on the training split of the AMASS [176] dataset. We follow UHC [172] and remove sequences that are noisy or involve interactions of human objects, resulting in 11313 high-quality training sequences and 140 test sequences. To evaluate our policy's ability to handle unseen MoCap sequences and noisy pose estimate from pose estimation methods, we use the popular H36M dataset [114]. From H36M, we derive two subsets *H36M-Motion** and *H36M-Test-Video**. H36M-Motion* contains 140 high-quality MoCap sequences from the entire H36M dataset. H36M-Test-Video* contains 160 sequences of noisy poses estimated from videos in the H36M test split (since SOTA pose estimation methods are trained on H36M's training split). * indicates the removal of sequences containing human-chair interaction.

Metrics. We use a series of pose-based and physics-based metrics to evaluate our motion imitation performance. We report the success rate (Succ) as in UHC [172], deeming imitation un-

successful when, at *any point* during imitation, the body joints are on average $> 0.5m$ from the reference motion. Succ measures whether the humanoid can track the reference motion without losing balance or significantly lags behind. We also report the root-relative mean per-joint position error (MPJPE) E_{mpjpe} and the global MPJPE $E_{\text{g-mpjpe}}$ (in mm), measuring our imitator's ability to imitate the reference motion both locally (root-relative) and globally. To show physical realism, we also compare acceleration E_{acc} (mm/frame^2) and velocity E_{vel} (mm/frame) difference between simulated and MoCap motion. All the baseline and our methods are physically simulated, so we do not report any foot sliding or penetration.

3.4.1 Motion Imitation

Motion Imitation on High-quality MoCap. Table3.1 reports our motion imitation result on the AMASS train, test, and H36M-Motion* dataset. Comparing with the baseline **with RFC**, our method outperforms it on almost all metrics across training and test datasets. On the training dataset, PHC has a better success rate while achieving better or similar MPJPE, showcasing its ability to better imitate sequences from the training split. On testing, PHC shows a high success rate on unseen MoCap sequences from both the AMASS and H36M data. Unseen motion poses additional challenges, as can be seen in the larger per-joint error. UHC trained without residual force performs poorly on the test set, showing that it lacks the ability to imitate unseen reference motion. Noticeably, it also has a much larger acceleration error because it uses high-frequency jitter to stay balanced. Compared to UHC, our controller has a low acceleration error even when facing unseen motion sequences, benefiting from the energy penalty and motion prior. Surprisingly, our keypoint-based controller is on par and sometimes outperforms the rotation-based one. This validates that the keypoint-based motion imitator can be a simple and strong alternative to the rotation-based ones.

Motion Imitation on Noisy Input from Video. We use off-the-shelf pose estimators HybrIK [143] and MeTRAbs [263] to extract joint rotation (HybrIK) and keypoints (MeTRAbs) using images from the H36M test set. As a post-processing step, we apply a Gaussian filter to the extracted pose and keypoints. Both HyBrIK and MeTRAbs are per-frame models that do not use any temporal information. Due to depth ambiguity, monocular global pose estimation is highly noisy [263] and suffers from severe depth-wise jitter, posing significant challenge to motion imitators. We find that MeTRAbs outputs better global root estimation $\tilde{\boldsymbol{p}}_t^0$, so we use its $\tilde{\boldsymbol{p}}_t^0$ combined with HybrIK's estimated joint rotation $\tilde{\boldsymbol{\theta}}_t$ (HybrIK + Metrabs (root)). In Table3.2, we report our controller and baseline's performance on imitating these noisy sequences. Similar to results on MoCap Imitation, PHC outperforms the baselines by a large margin and achieves a high success rate ($\sim 90\%$). This validates our hypothesis that PHC is robust to noisy motion and can be used to drive simulated avatars directly from videos. Similarly, we see that keypoint-based controller (ours-kp) outperforms rotation-based, which can be explained by 1) estimating 3D keypoint directly from images is an easier task than estimating joint rotations, so keypoints from MeTRABs are of higher quality than joint rotations from HybrIK; 2) our keypoint-based controller is more robust to noisy input as it has the freedom to use any joint configuration to try to match the keypoints.

H36M-Test-Video*							
RET	MCP	PNN	Rotation	Fail-Recover	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓
✗	✗	✗	✓	✗	51.2%	56.2	34.4
✓	✗	✗	✓	✗	59.4%	60.2	37.2
✓	✓	✗	✓	✗	66.2%	59.0	38.3
✓	✓	✓	✓	✗	86.9%	**53.1**	**33.7**
✓	✓	✓	✓	✓	88.7%	55.4	34.7
✓	✓	✓	✗	✓	**90.0%**	55.8	41.0

Table 3.3. Ablation on components of our pipeline, performed using noisy pose estimate from HybrIK + Metrabs (root) on the H36M-Test-Video* data. RET: relaxed early termination. MCP: multiplicative control policy. PNN: progressive neural networks.

Ablations. Table 3.3 shows our controller trained with various components disabled. We perform ablation on the noisy input from H36M-Test-Image* to better showcase the controller's ability to imitate noisy data. First, we study the performance of our controller before training to recover from fail-state. Comparing row 1 (R1) and R2, we can see that relaxed early termination (RET) allows our policy to better use the ankle and toes for balance. R2 vs R3 shows that using MCP directly without our progressive training process boosts the network performance due to its enlarged network capacity. However, using the PMCP pipeline significantly boosts robustness and imitation performance (R3 vs. R4). Comparing R4 and R5 shows that PMCP is effective in adding fail-state recovery capability **without** compromising motion imitation. Finally, R5 vs. R6 shows that our keypoint-based imitator can be on-par with rotation-based ones, offering a simpler formulation where only keypoints is needed.

Real-time Simulated Avatars. We demonstrate our controller's ability to imitate pose estimates streamed in real-time from videos. Fig.3.3 shows a qualitative result on a live demonstration of using poses estimated from an office environment. To achieve this, we use our keypoint-based controller and MeTRAbs-estimated keypoints in a streaming fashion. The actor performs a series of motions, such as posing and jumping, and our controller can remain stable. Fig.3.3 also shows our controller's ability to imitate reference motion generated directly from a motion language model MDM [277]. We provide extensive qualitative results in our website for our real-time use cases.

3.4.2 Fail-state Recovery

To evaluate our controller's ability to recover from fail-state, we measure whether our controller can successfully reach the reference motion within a certain time frame. We consider three scenarios: 1) fallen on the ground, 2) far away from reference motion, and 3) fallen and far from reference. We use a single clip of standing-still reference motion during this evaluation. We generate fallen-states by dropping the humanoid on the ground and applying random joint torques for 150 time steps. We create the far-state by initializing the humanoid 3 meters from the reference motion. Experiments are run randomly 1000 trials. From Tab.3.4 we can see that both of our keypoint-based and rotation-based controllers can recover from fall state with high

	Fallen-State		Far-State		Fallen + Far-State	
Method	Succ-5s ↑	Succ-10s ↑	Succ-5s ↑	Succ-10s ↑	Succ-5s ↑	Succ-10s ↑
Ours	95.0%	98.8%	83.7%	99.5%	93.4%	98.8%
Ours-kp	92.5%	94.6%	95.1%	96.0%	79.4%	93.2%

Table 3.4. We measure whether our controller can recover from the fail-states by generating these scenarios (dropping the humanoid on the ground & far from the reference motion) and measuring the time it takes to resume tracking.

success rate ($> 90\%$) even in the challenging scenario when the humanoid is both fallen and far away from the reference motion. For a more visual analysis of fail-state recovery, see our website.

3.5 Additional Details

Extensive qualitative results are provided on the project page. We highly encourage our readers to view them to better understand the capabilities of our method. Specifically, we show our method's ability to imitate high-quality MoCap data (both train and test) and noisy motion estimated from video. We also demonstrate real-time video-based (single- and multi-person) and language-based avatar (single- and multiple-clips) use cases. Lastly, we showcase our fail-state recovery ability.

3.6 Implementation Details

3.6.1 Training Details

Humanoid Construction. Our humanoid can be constructed from any kinematic structure, and we use the SMPL humanoid structure as it has native support for different body shapes and is widely adopted in the pose estimation literature. Fig.3.4 shows our humanoid constructed based on randomly selected gender and body shape from the AMASS dataset. The simulation result can then be exported and rendered as the SMPL mesh. We showcase two types of constructed humanoid: capsule-based and mesh-based. The capsule-based humanoid is constructed by treating body parts as simple geometric shapes (spheres, boxes, and capsules). The mesh-based humanoid is constructed following a procedure similar to SimPoE [329], where each body part is created by finding the convex hull of all vertices assigned to each bone. The capsule humanoid is easier to simulate and design, whereas the mesh humanoid provides a better approximation of the body shape to simulate more complex human-object interactions. We find that mesh-based and capsule-based humanoids do not have significant performance differences (see Sec.3.7) and conduct all experiments using the capsule-based humanoid. For a fair comparison with the baselines, we use the mean body shape of the SMPL with neutral gender for all evaluations and show qualitative results for shape variation. For both types of

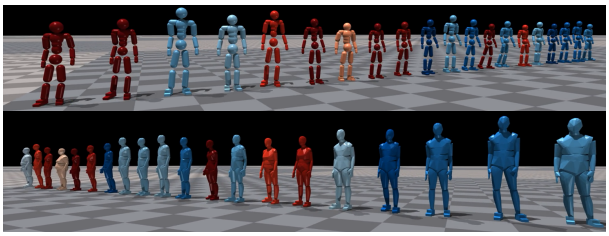


Figure 3.4. Our framework can support body shape and gender variations. Here we showcase humanoids of different gender and body proportion holding a standing pose. We construct two kinda of humanoids: capsule-based (top) and mesh-based (bottom). Red: female, Blue: male. Color gradient indicates weight.

	Batch Size	Learning Rate	σ	γ	ϵ
Value	1536	2×10^{-5}	0.05	0.99	0.2
	w_{jp}	w_{jr}	w_{jv}	$w_{\text{j}\omega}$	
Value	0.5	0.3	0.1	0.1	

Table 3.5. Hyperparameters for PHC. σ: fixed variance for policy. γ: discount factor. ϵ: clip range for PPO

humanoids, we scale the density of geometric shapes so that the body has the correct weight (on average 70 kg). All inter-joint collisions are enabled for all joint pairs except for between parent and child joints. Collision between humanoids can be enabled and disabled at will (for multi-person use cases).

Training Process. During training, we randomly sample motion from the current training set $\hat{\boldsymbol{Q}}^{(k)}$ and normalize it with respect to the simulated body shape by performing forward kinematics using $\hat{\boldsymbol{\theta}}_{1:T}$. Similar to UHC [172], we adjust the height of the root translation $\hat{\boldsymbol{p}}_t^0$ to make sure that each of the humanoid's feet touches the ground at the beginning of the episode. We use parallelly simulate 1536 humanoids for training all of our primitives and composers. Training takes around 7 days to collect approximately 10 billion samples. When training with different body shapes, we randomly sample valid human body shapes from the AMASS dataset and construct humanoids from them. Hyperparamters used during training can be found in Table.6.7

Data Preparation. We follow similar procedure to UHC [172] to filter out AMASS sequences containing human object interactions. We remove all sequences that sits on chairs, move on treadmills, leans on tables, steps on stairs, floating in the air *etc*., resulting in 11313 high-quality motion sequences for training and 140 sequences for testing. We use a heuristic-based filtering process based on *i.e*. identifying the body joint configurations corresponding to the sitting motion or counting number of consecutive airborne frames.

Runtime. Once trained, our PHC can run in real time ($\sim$ 32FPS) together with simulation and rendering, and around ($\sim$ 50FPS) when run without rendering. Table.3.6 shows the runtime of our method with respect to the number of primitives, architecture, and humanoid type used.

Model Size. The final model size (with four primitives) is 28.8 MB, comparable to the model size of UHC (30.4 MB).

3.6.2 Real-time Use Cases

Real-time Physics-based Virtual Avatars from Video. To achieve real-time physics-based avatars driven by video, we first use Yolov8 [124] for person detection. For pose estimation, we use MeTRAbS [263] and HybrIK [143] to provide 3D keypoints $\tilde{\boldsymbol{p}}_t$ and rotation $\tilde{\boldsymbol{\theta}}_t$. MeTRAbs is a 3D keypoint estimator that computes 3D joint positions $\tilde{\boldsymbol{p}}_t$ in the absolute global space (rather than in the relative root space). HybrIK is a recent method for human mesh recovery and computes joint angles $\tilde{\boldsymbol{\theta}}_t$ and root position $\tilde{\boldsymbol{p}}_t^0$ for the SMPL human body. One can recover the 3D keypoints $\tilde{\boldsymbol{p}}_t$ from joint angles $\tilde{\boldsymbol{\theta}}_t$ and root position $\tilde{\boldsymbol{p}}_t^0$ using forward kinematics. Both of these methods are causal, do not use any temporal information, and can run in real-time ($\sim$ 30FPS). Estimating 3D keypoint location from image pixels is an easier task than regressing joint angles, as 3D keypoints can be better associated with features learned from pixels. Thus, both HybrIK and MeTRAbs estimate 3D keypoints $\tilde{\boldsymbol{p}}_t$, with HybrIK containing an additional step of performing learned inverse kinematics to recover joint angles $\tilde{\boldsymbol{\theta}}_t$. We show results using both of these off-the-shelf pose estimation methods, using MeTRAbs with our keypoint-based controller and HybrIK with our rotation-based controller. Empirically, we find that MeTRAbs estimates more stable and accurate 3D keypoints, potentially due to its keypoint-only formulation. We also present a real-time **multi-person** physics-based human-to-human interaction use case, where we drive multiple avatars and enable inter-humanoid collision. To support multi-person pose estimation, we use OCSort [34] to track individual tracklets and associate poses with each person. Notice that real-time use cases pose additional challenges than offline processing: detection, pose/keypoint estimation, and simulation all need to run at real-time at around 30 FPS, and small fluctuations in framerate could lead to unstable imitation and simulation. To smooth out noisy depth estimates, we use a Gaussian filter to smooth out estimates from t-120 to t, and use the "mirror" setting for padding at boundary.

Virtual Avatars from Language. For language-based motion generation, we adopt MDM [277] as our text-to-motion model. We use the official implementation, which generates 3D keypoints $\tilde{\boldsymbol{p}}_t$ by default and connects it to our keypoint-based imitator. MDM generates fixed-length motion clips, so additional blending is needed to combine multiple clips of generated motion. However, since PHC can naturally go to far-away reference motion and handles disjoint between motion clips, we can naively chain together multiple clips of motion generated by MDM and create coherent and physically valid motion from multiple text prompts. This enables us to create a simulated avatar that can be driven by a continuous stream of text prompts.

3.6.3 Progressive Neural Network (PNN) Details

A PNN [242] starts with a single primitive network $\mathcal{P}^{(1)}$ trained on the full dataset $\hat{\boldsymbol{Q}}$. Once $\mathcal{P}^{(1)}$ is trained to convergence on the entire motion dataset $\hat{\boldsymbol{Q}}$ using the imitation task, we create a subset of hard motions by evaluating $\mathcal{P}^{(1)}$ on $\hat{\boldsymbol{Q}}$. Sequences that $\mathcal{P}^{(1)}$ fails forms $\hat{\boldsymbol{Q}}_{\text{hard}}^{(1)}$. We then freeze the parameters of $\mathcal{P}^{(1)}$ and create a new primitive $\mathcal{P}^{(2)}$ (randomly initialized) along with lateral connections that connect each layer of $\mathcal{P}^{(1)}$ to $\mathcal{P}^{(2)}$. Given the layer weight $\boldsymbol{W}_i^{(k)}$, activation function f, and the learnable lateral connection weights $\boldsymbol{U}_i^{(j:k)}$, we have the hidden activation $\boldsymbol{h}_i^{(k)}$ of the i^{th} layer of k^{th} primitive as:

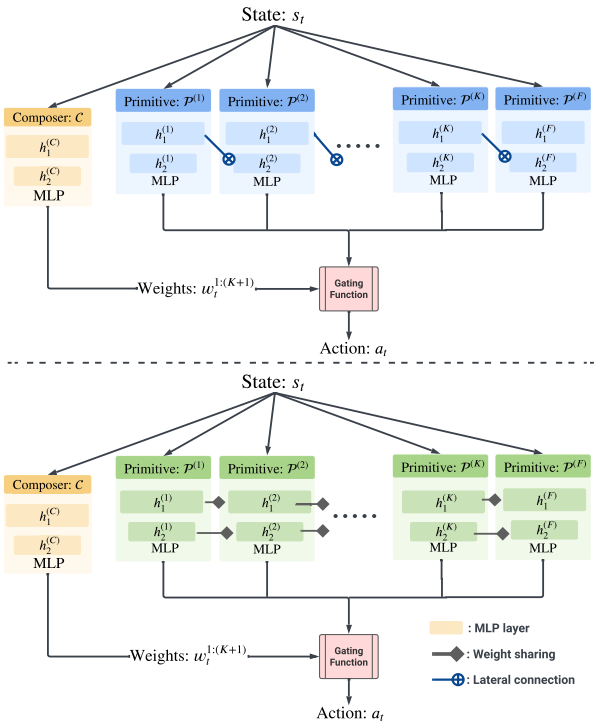


Figure 3.5. Progressive neural network architecture. Top: PNN with lateral connection. Bottom: PNN with weight sharing. $h_i^{(j)}$ indicates hidden activation of j^{th} primitive's i^{th} layer.

$$\boldsymbol{h}_i^{(k)} = f\left(\boldsymbol{W}_i^{(k)}\boldsymbol{h}_{i-1}^{(k)} + \sum_{j<k}\boldsymbol{U}_i^{(j:k)}\boldsymbol{h}_{i-1}^{(j)}\right). \tag{3.7}$$

Fig.3.5 visualizes the PNN with the lateral connection architecture. Essentially, except for the first layer, each subsequent layer receives the activation of the previous layer processed by the learnable connection matrices $\boldsymbol{U}_i^{(j:k)}$. We do not use any adapter layer as in the original paper. As an alternative to lateral connection, we explore weight sharing and warm-starts the primitive with the weights from the previous one (as opposed to randomly initialized). We find both methods equally effective (see Sec.3.7) when trained with the same hard-negative mining procedure, as each newly learned primitive adds new sequences that PHC can imitate. The weight sharing strategy significantly decreases training time as the policy starts learning harder sequences with basic motor skills. We use weight sharing in all our main experiments.

3.7 Supplementary Results

3.7.1 Categorizing the Forgetting Problem

As mentioned in the main paper, one of the main issues in learning to mimic a large motion dataset is the forgetting problem. The policy will learn new sequences while forgetting the ones already learned. In Fig.3.6, we visualize the sequences that the policy fails to imitate during training. Starting from the 12.5k epoch, each evaluation shows that some sequences are learned, but the policy will fail on some already learned sequences. The staircase pattern indicates that when learning sequences failed previously,

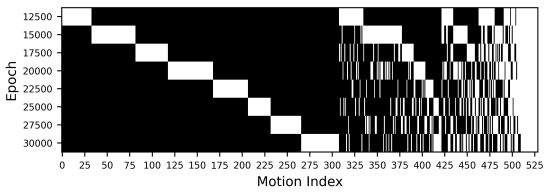


Figure 3.6. Here we plot the motion indexes that the policy fails on over training time; we only plot the 529 sequences that the policy has failed on over these training epoches. A white pixel denotes that sequence is can be successfully imitated at the given epoch, and a black pixel denotes an unsuccessful imitation. We can see that while there are 30 sequences that the policy consistently fails on, the remaining can be learned and then forgotten as training progresses. The staircase pattern indicates that the policy fails on new sequences each time it learns new ones.

the policy forgets already learned sequences. Numerically, each evaluation has around 30% overlap of failed sequences (right end side). The 30% overlap contains the backflips, cartwheeling, and acrobatics; motions that the policy consistently fails to learn when trained together with other sequences. We hypothesize that these remaining sequences (around 40) may require additional sequence-level information for the policy to learn properly together with other sequences.

Fail-state recovery Learning the fail-state recovery task can also lead to forgetting previously learned imitation skills. To verify this, we evaluate $\boldsymbol{\mathcal{P}}^{(F)}$ on the H36M-Test-Video dataset, which leads to a performance of Succ: 42.5%, $E_{\text{g-mpjpe}}$: 87.3, and E_{mpjpe}: 55.9, which is much lower than the single primitive $\boldsymbol{\mathcal{P}}^{(1)}$ performance of Succ: 59.4%, $E_{\text{g-mpjpe}}$: 60.2, and E_{mpjpe}: 34.4. Thus, learning the fail-state recovery task may lead to severe forgetting of the imitation task, motivating our PMCP framework to learn separate primitives for imitation and fail-state recovery.

3.7.2 Additional Ablations

In this section, we provide additional ablations of the components of our framework. Specifically, we study the effect of MOE vs. MCP, lateral connection vs. weight sharing, and the number of primitives used. We also report the inference speed (counting network inference and simulation time). All experiments are carried out with the rotation-based imitator and incorporate the fail state recovery primitive $\boldsymbol{\mathcal{P}}^{(F)}$ as the last primitive.

PNN Lateral Connection vs. Weight Sharing. As can be seen in Table 3.6, comparing Row 1 (R1) and R7 , we can see that PNN with lateral connection and weight sharing produce similar performance, both in terms of motion imitation and inference speed. This shows that *in our setup*, the weight sharing scheme is an effective alternative to lateral connections. This can be explained by the fact that in our case, each "task" on which the primitives are trained is similar and does not require lateral connection to choose whether to utilize prior experiences or not.

MOE vs. MCP. The difference between the top-1 mixture of experts (MOE) and multiplicative control (MCP) is discussed in detail in the MCP paper [212]: top-1 MOE only activates one expert at a time, while MCP can activate all primitives at the same time. Comparing R2 and R7, as expected, we can see that top-1 MOE is slightly inferior to MCP. Since all of our primitives are pretrained and frozen, theoretically a perfect composer should be able to choose the best primitive based on input for both MCP and MOE. MCP, compared to MOE, can activate all primitives at once and search a large action space where multiple primitives can be combined. Thus, MCP provides better performance, while MOE is not far behind. This is also observed by CoMic [93], where they observe similar performance between mixture and product distributions when used to combine subnetworks. Note that top-inf MOE is similar

						H36M-Test-Video*			
PNN-Lateral	PNN-Weight	MOE	MCP	Type	# Prim	Succ ↑	$E_{\text{g-mpjpe}} \downarrow$	$E_{\text{mpjpe}} \downarrow$	FPS
✓	✗	✗	✗	Cap	4	87.5%	55.7	36.2	32
✗	✓	✓	✗	Cap	4	87.5%	56.3	34.3	33
✗	✓	✗	✓	Mesh	4	86.9%	62.6	39.5	30
✗	✗	✗	✗	Cap	1	59.4%	60.2	37.2	32
✗	✓	✗	✓	Cap	2	65.6%	58.7	37.3	32
✗	✓	✗	✓	Cap	3	80.9%	56.8	36.1	32
✗	✓	✗	✓	Cap	4	**88.7%**	**55.4**	**34.7**	32
✗	✓	✗	✓	Cap	5	87.5%	57.7	36.0	32

Table 3.6. Supplmentary ablation on components of our pipeline, performed using noisy pose estimate from HybrIK + Metrabs (root) on the H36M-Test-Video* data. MOE: top-1 mixture of experts. MCP: multiplicative control policy. PNN: progressive neural networks. Type: between Cap (capsule) and mesh-based humanoids. All models are trained with the same procedure.

to MCP where all primitives can be activated.

Capsule vs. Mesh Humanoid. Comparing R3 and R7, we can see that mesh-based humanoid yield similar performance to capsule-based ones. It does slow down simulation by a small amount (30 FPS vs. 32 FPS), as simulating mesh is more compute-intensive than simulating simple geometries like capsules.

Number of primitives. Comparing R4, R5, R6, R7, and R8, we can see that the performance increases as the number of primitives increases. Since the last primitive $\mathcal{P}^{(F)}$ is for fail-state recovery and does not provide motion imitation improvement, R5 is similar to the performance of models trained without PMCP (R4). As the number of primitives grows from 2 to 3, we can see that the model performance grows quickly, showing that MCP is effective in combining pretrained primitives to achieve motion imitation. Since we are using relatively small networks, the inference speed does not change significantly with the number of primitives used. We notice that as the number of primitives grows, $\hat{\boldsymbol{Q}}^{(k)}$ becomes more and more challenging. For instance, $\hat{\boldsymbol{Q}}^{(4)}$ contains mainly highly dynamic motions such as high-jumping, back flipping, and cartwheeling, which are increasingly difficult to learn together. We show that (see supplementary webpage) we can overfit these sequences by training on them only, yet it is significantly more challenging to learn them together. Motions that are highly dynamic require very specific steps to perform (such as moving while airborne to prepare for landing). Thus, the experiences collected when learning these sequences together may contradict each other: for example, a high jump may require a high speed running up, while a cartwheel may require a different setup of foot-movement. A per-frame policy that does not have sequence-level information may find it difficult to learn these sequences together. Thus, sequence-level or information about the future may be required to learn these high dynamic motions together. In general, we find that using 4 primitives is most effective in terms of training time and performance, so for our main evaluation and visualizations, we use **4-primitive models**.

3.8 Discussions

Limitations. While our purposed PHC can imitate human motion from MoCap and noisy input faithfully, it does not achieve a 100% success rate on the training set. Upon inspection, we find that highly dynamic motions such as high jumping and back flipping are still challenging. Although we can train

single-clip controller to **overfit** on these sequences (see the website), our full controller often fails to learn these sequences. We hypothesize that learning such highly dynamic clips (together with simpler motion) requires more planning and intent (*e.g.* running up to a high jump), which is not conveyed in the single-frame pose target $\hat{\boldsymbol{q}}_{t+1}$ for our controller. The training time is also long due to our progressive training procedure. Furthermore, to achieve better downstream tasks, the current disjoint process (where the video pose estimator is unaware of the physics simulation) may be insufficient; tighter integration with pose estimation [173, 329] and language-based motion generation [328] is needed.

As discussed in the main paper, PHC has yet to achieve 100% success rate on the AMASS training set. With a 98.9% success rate, PHC can imitate *most* of our daily motion without losing balance, but can still struggle to perform more dynamic motions, such as backflipping. For our real-time avatar use cases, we can see a noticeable degradation in performance from the offline counterparts. This is due to the following:

- Discontinuity and noise in reference motion. The inherent ambiguity in monocular depth estimation can result in noisy and jittery 3D keypoints, particularly in the depth dimension. These small errors, though sometimes imperceptible to the human eye, may provide PHC with incorrect movement signals, leaving insufficient time for appropriate reactions. Velocity estimation is also especially challenging in real-time use cases, and PHC relies on stable velocity estimation to infer movement cues.
- Mismatched framerate. Since our PHC assumes 30 FPS motion input, it is essential for pose estimates from video to match for a more stable imitation. However, few pose estimators are designed to perform real-time pose estimation ($\geq$ 30 FPS), and the estimation framerate can fluctuate due to external reasons, such as the load balance on computers.
- For multi-person use case, tracking and identity switch can still happen, leading to a jarring experience where the humanoids need to switch places.

A deeper integration between the pose estimator and our controller is needed to further improve our real-time use cases. As we do not explicitly account for camera pose, we assume that the webcam is level with the ground and does not contain any pitch or roll. Camera height is manually adjusted at the beginning of the session. The pose of the camera can be taken into account in the pose estimation stage. Another area of improvement is naturalness during fail-state recovery. While our controller can recover from fail-state in a human-like fashion and walks back to resume imitation, the speed and naturalness could be further improved. Walking gait, speed, and tempo during fail-state recovery exhibits noticeable artifacts, such as asymmetric motion, a known artifact in AMP [215]. During the transition between fail-state recovery and motion imitation, the humanoid can suddenly jolt and snap into motion imitation. Further investigation (*e.g.* better reward than the point-goal formulation, additional observation about trajectory) is needed.

Conclusion and Future Work. We introduce Perpetual Humanoid Controller, a general purpose physics-based motion imitator that achieves high quality motion imitation while being able to recover from fail-states. Our controller is robust to noisy estimated motion from video and can be used to perpetually simulate a real-time avatar without requiring reset. Future directions include 1) improving imitation capability and learning to imitate 100% of the motion sequences of the training set; 2) incorporating terrain and scene awareness to enable human-object interaction; 3) tighter integration with downstream tasks such as pose estimation and motion generation, *etc*. Given reference motion, PHC can provide the mobility for the humanoid to move. In the next chapter, we will discuss how to use PHC's capabilities for controlling simulated humanoids from visual input.

Chapter 4

SimXR: Real-Time Simulated Avatar from Head-Mounted Sensors

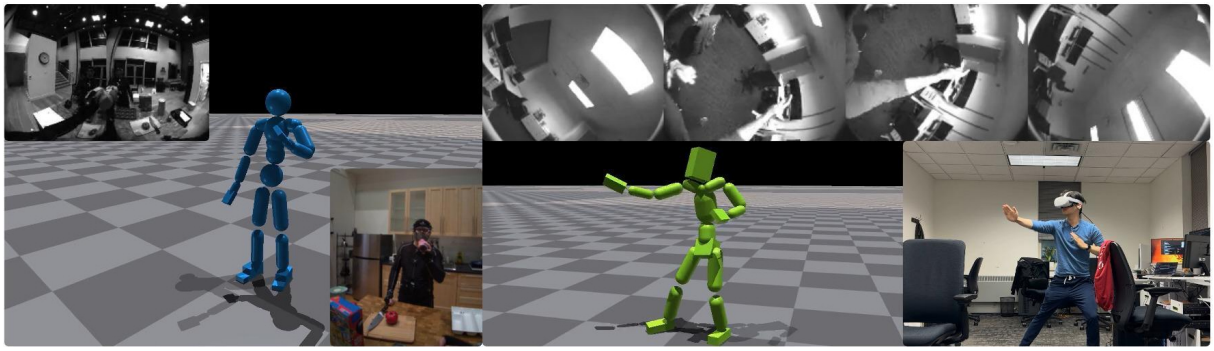


Figure 4.1. Avatar control using *PULSE* on real world AR/VR headsets. (*Left*): An indoor kitchen setting using AR headset. PULSE controls the humanoid using headset pose and visual input from two front-facing cameras. (*Right*): An office setting using VR headset (Quest 2). Humanoid motion is driven by the headset pose, two side-facing and two up-facing cameras.

4.1 Introduction

In this chapter, equipped with PHC, we propose an alternative to the "kinematic policy" used in Chapter 2. Instead of using the imitator in the loop, we distill its motor skills into an end-to-end, "photons-to-actions" policy for the egocentric pose estimation task.

From the sensor streams captured by a head-mounted device (AR / VR, or XR headsets), we aim to control a simulated humanoid / avatar to follow the wearer's global 3D body pose in real-time, as shown in Fig.4.1. This could be applied to animating virtual avatars in mixed reality, games, and potentially teleoperating humanoid robots [98]. However, the sensor suite of commercially available head-mounted devices is rarely designed for full-body pose estimation. Their cameras are often facing forward (*e.g.* Aria glasses [257]) or on the side (*e.g.* Meta Quest [4]) and are used mainly for Simultaneous Location and Mapping (SLAM) and hand tracking. Thus, the body is seen from extreme and distorted viewpoints from these cameras.

These challenges have led to research on vision-based egocentric pose estimation to create scenarios

with more favorable camera placement (*e.g.* fisheye cameras directly pointing downward [7, 280, 288, 289, 309, 342]), where more body parts can be observed. These camera views are often unrealistic and hard to recreate in the real-world: a camera protruding out and facing downward could be out of the budget or break the aesthetics of the product. As there is no standard for these heterogeneous camera specifications, it is difficult to collect large-scale data. Using synthetic data [7, 280] could alleviate this problem to some extent, but real-world recreation of the camera specification used in rendering is still challenging, exacerbating the sim-to-real gap.

Another line of work instead uses head tracking to infer full-body motion. The 6 degree-of-freedom (DoF) headset pose, being considerably lower in dimensionality compared to images, is easily accessible from extended reality (XR) headsets. However, the headset pose alone contains insufficient information on full body movement, so previous work either formulates the task as a generative one [140] or relies on action labels [172] to further constrain the solution space. Adding VR controllers as input can provide additional information on hand movement and can lead to a more stable estimate of full body pose, but the VR controller is not always available [1, 11, 217], especially for light-weight AR glasses with forward-facing cameras. These methods are also primarily kinematics-based [10, 11, 67, 122], focusing on motion estimation without taking into account the underlying forces. As a result, they cause floating and penetration problems, especially because feet are often unobserved.

To combat these issues, physics simulation [295] and environmental cues [138] have been used to create plausible foot motion. Leveraging the laws of physics can significantly improve motion realism and force the simulated character to adopt a viable foot movement. However, incorporating physics introduces the additional challenge of humanoid control–humanoids need to be balanced and track user movement at all times. Most physics-based methods are learned using reinforcement learning (RL) and require millions (sometimes billions) of environment interactions. If each interaction requires processing of the raw image input, the computational cost would be significant. As a result, approaches that use vision and simulated avatars often use a low-dimensional intermediate representation, such as precomputed image features [325] or kinematic poses [172, 173, 329]. Although a controller can be trained to consume these intermediate features, this approach creates a disconnect between the visual and control components, where the visual component does not receive adequate feedback during training from the physics simulation.

In this work, we demonstrate the feasibility of an end-to-end simulated avatar control framework for XR headsets. Our approach, ***SIM**ulated Avatar from **XR** sensors* (SimXR), directly maps the input signals to joint actuation without relying on intermediate representations such as body pose or 2D keypoints. Our key design choice is to distill from a pre-trained motion imitator to learn the mapping from input to control signals, which enables efficient learning from vision input. Due to its simplistic design and learning framework, our approach is compatible with a diverse selection of smart headsets, ranging from VR goggles to lightweight AR headsets (as shown in Fig.4.2). Since no dataset exists for the camera configurations of commercially available VR headsets, we also propose a large-scale synthetic dataset (2216k frames) and a real-world dataset (40k frames) for testing and show that our method can be applied to real-world and real-time use cases.

To summarize, our contributions are: (1) we design a method to use simulated humanoids to estimate global full-body pose using images and headset pose from an XR headset (front-facing AR cameras or side-facing VR ones); (2) we demonstrate the feasibility of learning an end-to-end controller to directly map from input sensor features to control signals through distillation; (3) we contribute large-scale synthetic and real-world datasets with commercially available VR headset configuration for future research.

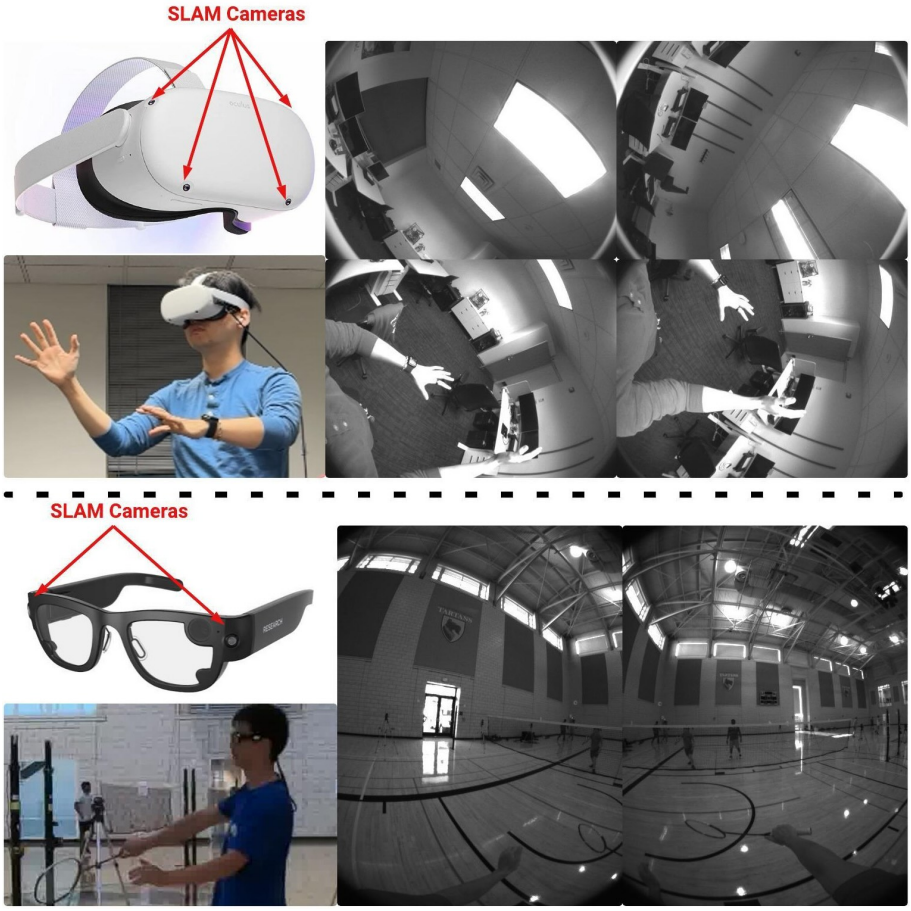


Figure 4.2. SimXR framework applied to two AR/VR devices. (*Top*): Quest 2 [3] headset with 4 SLAM cameras, two facing upward and two downward. (*Bottom*): Aria glass [2] with two forward-facing SLAM cameras. Both devices provides 6DoF headset tracking in real-time.

4.2 Related Work

Pose Estimation from Head-Mounted Sensors. Due to the lack of dataset using commercially available devices, egocentric pose estimation has been studied using synthetic data [7, 280], and small-scale real-world data captured from custom camera rigs [121, 235, 289, 309, 324]. Among them, EgoCap [235] uses two downward facing cameras protruding from the helmet, while Jiang *et al*. [121] uses a chest-mounted camera. Later, Mo2Cap2 [309] and SelfPose [280] use a head-mounted downward-facing fisheye camera for pose estimation in the camera's coordinate system. All use synthetic data for training, while Mo2Cap2 also captures a real-world dataset (10k frames) for testing. Wang *et al*. [288, 289] extends a similar setup for global 3D body pose estimation by extracting head movement from video and optimization-based global pose refinement. To combat the lack of large-scale dataset, UnrealEgo [7] uses a dual fish-eye camera setup to generate synthetic data using a game engine. All of the above settings have cameras that directly point downward at the human body. However, cameras on commercially available XR headsets often do not have a dedicated body camera and only have monochrome SLAM or hand-tracking cameras. For VR devices, these cameras point to the side and have very limited visibility of the hands and feet. For AR glasses, cameras often point forward and provide only fleeting hand visibility. Due to challenging viewpoints, some work uses head tracking as an alternative [140, 172, 217, 295, 295, 324, 325] for pose

estimation. Among them, Egopose [325] estimates locomotion, while KinPoly [172] extends it to action-conditioned pose estimation. EgoEgo [140] proposes the first general-purpose pose estimator that uses only the head pose. While they utilize front-facing images, the images are used to extract the head pose, rather than to provide information about the body pose. In this work, we take advantage of both camera views and headset pose for full-body avatar control.

Simulated Humanoid Motion Imitation. Motion imitation is an important humanoid control task that has seen steady progress in recent years [9, 46, 79, 169, 208, 212, 292, 297, 327]. Since no ground-truth data exist of human joint actuation and physics simulators are often nondifferentiable, a policy / imitator / controller is often trained to track / imitate / mimic human motion using deep reinforcement learning (RL). Although methods such as SuperTrack [79] and DiffMimic [234] explore more efficient ways than RL to train imitators, learning a robust policy to track a large amount of human motion remains challenging. Nonetheless, from policies that can track a single clip of motion [208], to large-scale datasets [169, 297], the applicability of motion imitators to downstream tasks grows. Previously, UHC [172], a motion imitator based on an external non-physical force [327], has been used for egocentric [172] and third-person scene-aware [173] pose estimation. Its follow-up, PHC [169], removes the dependency on the non-physical force.

Simulated Humanoid Control for Pose Estimation. Our work follows recent advances [84, 88, 108, 173, 291, 329] in using the laws of physics as a strong prior to estimating full-body motion. Mapping directly from images to humanoid control signals is quite challenging due to the complex dynamics of a humanoid and the diversity in natural images. Training motion controllers using RL is also notoriously sample inefficient, forcing some vision-based robotic approaches to use distillation [348] or very small images [184]. Therefore, most physics-based methods separate the problem into two distinct components: image-based pose estimation and humanoid motion imitation. First, the pose is estimated from the images using an off-the-shelf body pose [132] or a keypoint detector [81]. The estimated pose is then fed to a pre-trained imitator for further refinement [173, 329], sampling-based control [108], or co-training [84]. Some methods also employ trajectory optimization [232, 250, 251, 303], but the optimization process can be time-consuming, unless certain compromises are assumed (*e.g.* applying external non-physical forces [250, 251]). This disjoint process uses the kinematic body pose as an intermediate representation to communicate movement information to the humanoid controller. However, this communication layer can be fragile and adversely affected by the imitator's sensitivity to the intermediate representation. Thus, we propose to remove the kinematic pose layer and directly learn a mapping from the input to the control signals end-to-end.

4.3 Approach

At each time step, given the images $\boldsymbol{I}_t$ and the 6DoF pose ${\boldsymbol{q}_t}^{\tau}$ captured by the headset, our task is to drive a simulated avatar to match the full body pose of the camera wearer $\boldsymbol{q}_t$. We use monochrome SLAM cameras on XR headsets, producing images of dimension $\boldsymbol{I}_t \in \mathbb{R}^{V \times H \times W \times C}$ of V views (2 views for Aria glasses, 4 for Quest) and $C = 1$ channels for monochrome images. In Sec. 4.3.1, we briefly describe our synthetic data generation pipeline. In Sec. 4.3.2, we describe our proposed method SimXR and how to learn this controller.

A trained motion imitator could be used as a teacher and distill its motor skills for downstream tasks [24, 170, 184, 299], and we follow this paradigm and use PHC as a teacher to learn the mapping between the XR headset sensor and the control signals.

Avatar Control using head-mounted Sensors. Following the above definition, the task of controlling a simulated humanoid to match egocentric observation from head-mounted sensors can be formulated as using a control policy $\pi_{\text{SimXR}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau})$ to compute the joint action $\boldsymbol{a}_t$ based on image $\boldsymbol{I}_t$, headset

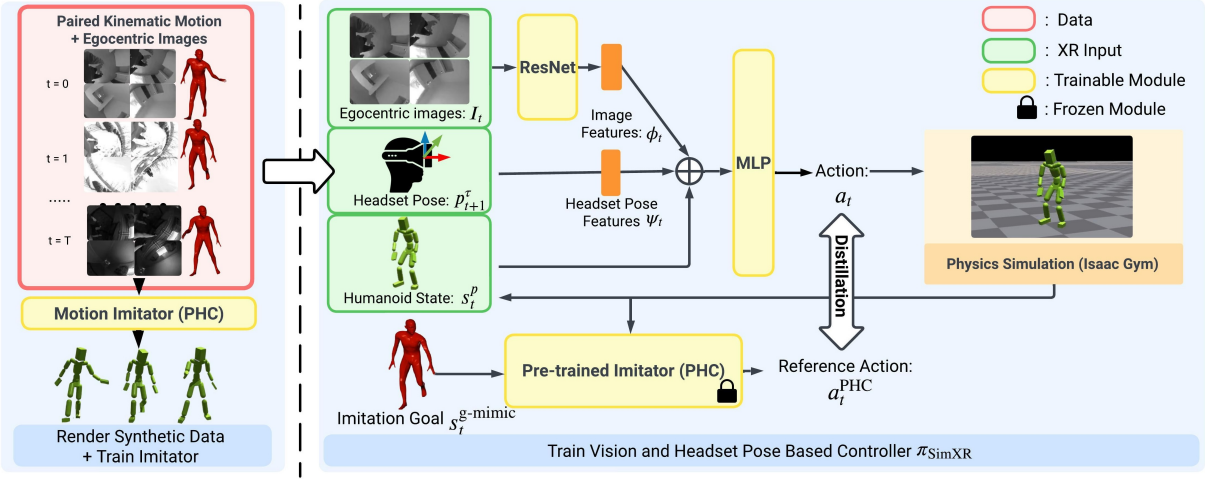


Figure 4.3. Our proposed SimXR framework. From a large-scale human motion dataset, we first train a motion imitator (PHC [169]) and render synthetic images. Then, we train our vision and headset pose-based controller through distilling from the pretrained imitator.

pose $\boldsymbol{q}^{\tau}_{t+1}$, and humanoid proprioception $\boldsymbol{s}^{\text{p}}_t$ to match the headset wearer's body pose $\hat{\boldsymbol{q}}_{1:T}$.

4.3.1 Synthetic Data for XR Avatar

As paired motion data and XR cameras & headset tracking is difficult to obtain, we use synthetic data to train our method when real data is not available. Specifically, since no data exist for Quest 2's SLAM camera configuration, we create a large-scale synthetic one. We render human motion from a large-scale internal MoCap dataset using the exact camera placement and intrinsic of the Quest 2 headset in the Unity game engine [21]. The headset moves with the head movement of the wearer as if it is worn. Our motion dataset contains diverse poses ranging from daily activities to jogging, stretching, gesturing, sports, *etc*., similar to AMASS [176] but uses a different kinematic structure from SMPL [161]. During rendering, we randomize clothing, lighting, and background images (projected onto a sphere) for **every** frame for domain randomization. As a result, the methods trained using our synthetic data can be applied to real-world scenarios without additional image-based domain augmentation during training. We render RGB images and then convert them to monochrome. Fig. 4.3 shows samples of our synthetic data. For more information on our synthetic data, please refer to the Supplement.

4.3.2 Simulated Humanoid Control From Head-Mounted Sensors

Sensor Input Processing. At each frame, the head-mounted sensors provide the image $\boldsymbol{I}_{t+1}$ and the 6DoF headset pose $\boldsymbol{p}^{\tau}_{t+1}$ (here we use t+1 to indicate the incoming pose/image for tracking). The input image $\boldsymbol{I}_{t+1}$ (which contains all monochrome views) is first processed with a lightweight image feature extractor $\mathcal{F}$ (*e.g*. ResNet18 [95]) to compute image features: $\phi_t = \mathcal{F}(\boldsymbol{I}_{t+1})$. All V camera views share the same feature extractor (Siamese network). We also replace all batch normalization layers [113] with group normalization [302] for training stability.

For the 6DoF headset pose ${\boldsymbol{q}_t}^{\tau}$, we treat it as a virtual "joint", where $\boldsymbol{q}^{\tau}_{t+1} \triangleq (\boldsymbol{p}^{\tau}_{t+1}, \boldsymbol{\theta}^{\tau}_{t+1})$ contains the global rotation $\boldsymbol{\theta}^{\tau}_{t+1}$ and translation $\boldsymbol{p}^{\tau}_{t+1}$ of the headset. We use the humanoid head to track the pose of the headset by computing the rotation and translation difference between the head joint ${\boldsymbol{q}_t}^{\text{Head}}$ and the headset pose $\boldsymbol{q}^{\tau}_{t+1}$. This creates the headset pose feature: $\psi_t = ({\boldsymbol{\theta}_t}^{\text{Head}} \ominus \boldsymbol{\theta}^{\tau}_{t+1}, {\boldsymbol{p}_t}^{\text{Head}} - \boldsymbol{p}^{\tau}_{t+1})$. The image

Algo 4: Learn SimXR via distillation

Input: Ground truth paired XR sensor input and pose dataset $\hat{\boldsymbol{Q}}$, pretrained PHC π_{PHC} ;
while *not converged* **do**
  $\boldsymbol{M} \leftarrow \emptyset$ initialize sampling memory ;
  while $\boldsymbol{M}$ *not full* **do**
    $\hat{\boldsymbol{q}}_{1:T}, \boldsymbol{I}_{1:T}, \boldsymbol{s}_t^{\text{p}}, \leftarrow$ sample motion, images and initial state from $\hat{\boldsymbol{Q}}$;
    for $t \leftarrow 1...T$ **do**
      $\boldsymbol{a}_t \leftarrow \pi_{\text{PULSE}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau})$;
                                  // compute humanoid action;
      $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ // simulation;
      $\boldsymbol{s}_t^{\text{g-mimic}} \leftarrow \mathcal{G}(\boldsymbol{s}_t^{\text{p}}, \hat{\boldsymbol{q}}_{t+1})$;
                          // compute imitation target for PHC;
      store $\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau}$ into memory $\boldsymbol{M}$;
  $\boldsymbol{a}_t^{\text{PHC}} \leftarrow \pi_{\text{PHC}}(\boldsymbol{a}_t^{\text{PHC+}} | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ annotate collected states in $\boldsymbol{M}$ using π_{PHC} ;
  $\pi_{\text{PULSE}} \leftarrow$ supervised update for π_{PULSE} using pairs of $(\boldsymbol{a}_t, \boldsymbol{a}_t^{\text{PHC}})$ and Eq.4.1.

feature and the headset pose features are then concatenated with proprioception $\boldsymbol{s}_t^{\text{p}}$ to form the input to an MLP to compute joint actions, as illustrated in Fig.4.3.

Online Distillation. Learning to control humanoids using RL requires a large number of simulation steps (*e.g.* billions of environment interactions), and our early experiments show that directly training an image-based policy using RL is infeasible (see Sec. 4.4.2). This is due to the large increase in input and model size (*e.g.* four 120×160 monochrome images versus 938d imitation goal $\boldsymbol{s}_t^{\text{g-mimic}}$) that prohibitively slows down training steps. Therefore, we opt for online distillation and use a pre-trained motion imitator, PHC π_{PHC}, to teach our policy π_{SimXR} via supervised learning. In short, we offload the sample-inefficient RL training to a low-dimensional state task (motion imitation) and use sample-efficient supervised learning for the high-dimensional image processing task. This is similar to the process proposed in PULSE [170] with the important distinction that PULSE uses the same input and output for the student and teacher, while ours has drastically different input modalities (images and headset pose vs. 3D pose).

Concretely, we train our pose estimation policy π_{SimXR} following the standard RL training framework: for each episode, given paired egocentric images $\boldsymbol{I}_{1:T}$, headset pose $\boldsymbol{q}_{1:T}^{\tau}$, and the corresponding reference full-body pose $\hat{\boldsymbol{q}}_{1:T}$, the humanoid is first initialized as $\hat{\boldsymbol{q}}_0$. Then, the policy $\pi_{\text{SimXR}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{I}_t, \boldsymbol{q}_{t+1}^{\tau})$ computes the joint actuation for the forward dynamics computed by physics simulation. By rolling out the policy in simulation, we obtain paired trajectories of $(\boldsymbol{s}_{1:T}^{\text{p}}, \boldsymbol{I}_{1:T}, \boldsymbol{q}_{1:T}^{\tau}, \hat{\boldsymbol{q}}_{1:T})$. Using the ground truth reference pose $\hat{\boldsymbol{q}}_{1:T}$ and simulated humanoid states $\boldsymbol{s}_{1:T}^{\text{p}}$, we can compute the per-frame imitation target for our pretrained imitator $\boldsymbol{s}_t^{\text{g-mimic}} \leftarrow \mathcal{G}(\boldsymbol{s}_t^{\text{p}}, \hat{\boldsymbol{q}}_{t+1})$. Then, using paired $(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$, we query PHC $\pi_{\text{PHC}}(\boldsymbol{a}_t^{\text{PHC}} | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ to calculate the reference action $\boldsymbol{a}_t^{\text{PHC}}$. This is similar to DAgger [239], where we use an expert to annotate reference actions for the student to learn from. To update π_{SimXR}, the loss is:

$$\mathcal{L} = \|\boldsymbol{a}_t^{\text{PHC}} - \boldsymbol{a}_t\|_2^2, \tag{4.1}$$

using standard supervised learning. In this way, our policy is trained end-to-end, where the image feature extractors $\mathcal{F}$ and the policy networks are directly updated using the gradient of $\mathcal{L}$. The learning process is also described in Alg. 4.

4.4 Experiments

Humanoids. Due to the different annotation formats between our proposed synthetic dataset and public

Aria Glasses			Quest 2				
Aria Digital Twin [200]			Synthetic			Real-world	
Train	Test	Annot.	Train	Test	Annot.	Test	Annot.
242 / 94k	64 /25k	MoCap	4097 / 1758k	1072/458k	MoCap	10/40k	Mono

Table 4.1. Dataset statistics and annotation source. MoCap: motion capture; Mono: monocular third-person pose estimation.

			Synthetic-Test						Real-world			
Method	Size	Physics	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$ ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{pa-mpjpe}}$ ↓	E_{acc} ↓	E_{vel} ↓
UnrealEgo [7]	554.5MB	✗	-	-	**56.9**	47.2	54.5	32.5	-	81.0	46.7	35.1
KinPoly-v [172]	98.9MB	✓	82.2%	66.7	63.5	42.8	**4.4**	**5.8**	9/10	83.0	7.0	11.2
Ours	59.7MB	✓	**94.3%**	**66.4**	62.4	**40.0**	6.5	8.3	**10/10**	**73.0**	**6.8**	**10.5**

Table 4.2. Pose estimation result on the test split (458k frames) of synthetic data and real-world captures (40k frames). Here, our MPJPE is computed as "device-relative" instead of root-relative.

datasets, we use two different humanoids for our experiments, one for the Quest VR headset and one for Aria AR glasses. For Aria, we use a humanoid following the SMPL [161] kinematic structure with the mean shape. It has 24 joints, of which 23 are actuated, resulting in an actuation space of $\mathbb{R}^{23\times3}$. Each degree of freedom is actuated by a proportional derivative (PD) controller, and the action $\boldsymbol{a}_t$ specifies the PD target. For the VR datasets, we use a humanoid that has 25 joints (out of which 24 are actuated). The imitator for SMPL-humanoid is trained on the AMASS [176] dataset, while for the in-house humanoid it is trained on the same in-house motion capture used to create our synthetic dataset.

Datasets. To train our control policies, we require high-quality motion data paired with camera views, as training with low-quality data might lead to the simulated character picking up unwanted behavior. Thus, for pose estimation using the Quest headset, we train solely on synthetic data created using a large-scale in-house motion capture dataset. We randomly split the data using an 8:2 ratio, resulting in 1758k frames for training and 458k frames for testing. To test the performance of our method in real-world scenarios, we also collect a real-world dataset containing 40k frames recorded by three different subjects. This dataset contains paired headset poses, SLAM camera images, and third-person images. We use the third-person images to create pseudo-ground truth using a SOTA monocular pose estimation method [262]. The real-world dataset contains daily activity motion common in VR scenarios, such as hand movements, boxing, kicking, *etc*. For pose estimation using Aria glasses, we use the recently proposed Aria Digital Twin dataset (ADT) [200]. The ADT dataset contains indoor motion sequences captured using MoCap suits and 3D scene scanners. It contains 119k frames that have paired skeleton and AR headset sensor output, recorded in a living room environment. This dataset only contains 3D keypoint annotations (no rotation), and we fit the SMPL body annotation to the 3D keypoints using a process similar to 3D-Simplify [23, 204]. We randomly split the ADT dataset for training (94k frames) and testing (25k frames). We do not use synthetic data for training the AR controller since there are available real-world ground-truth data.

Metrics. We report both pose- and physics-based metrics to evaluate our avatar's performance. We report the success rate (Succ) as in UHC [172], defined as: at *every point* during imitation, the head joint is < 0.5m from the headset pose. For pose estimation, we report the **device-relative** (instead of root relative) per-joint position error (MPJPE) E_{mpjpe}, global MPJPE $E_{\text{g-mpjpe}}$, and MPJPE after Procrustes analysis $E_{\text{pa-mpjpe}}$. Since $E_{\text{pa-mpjpe}}$ solves for the best matching scale, rotation, and translation, it is more suitable for

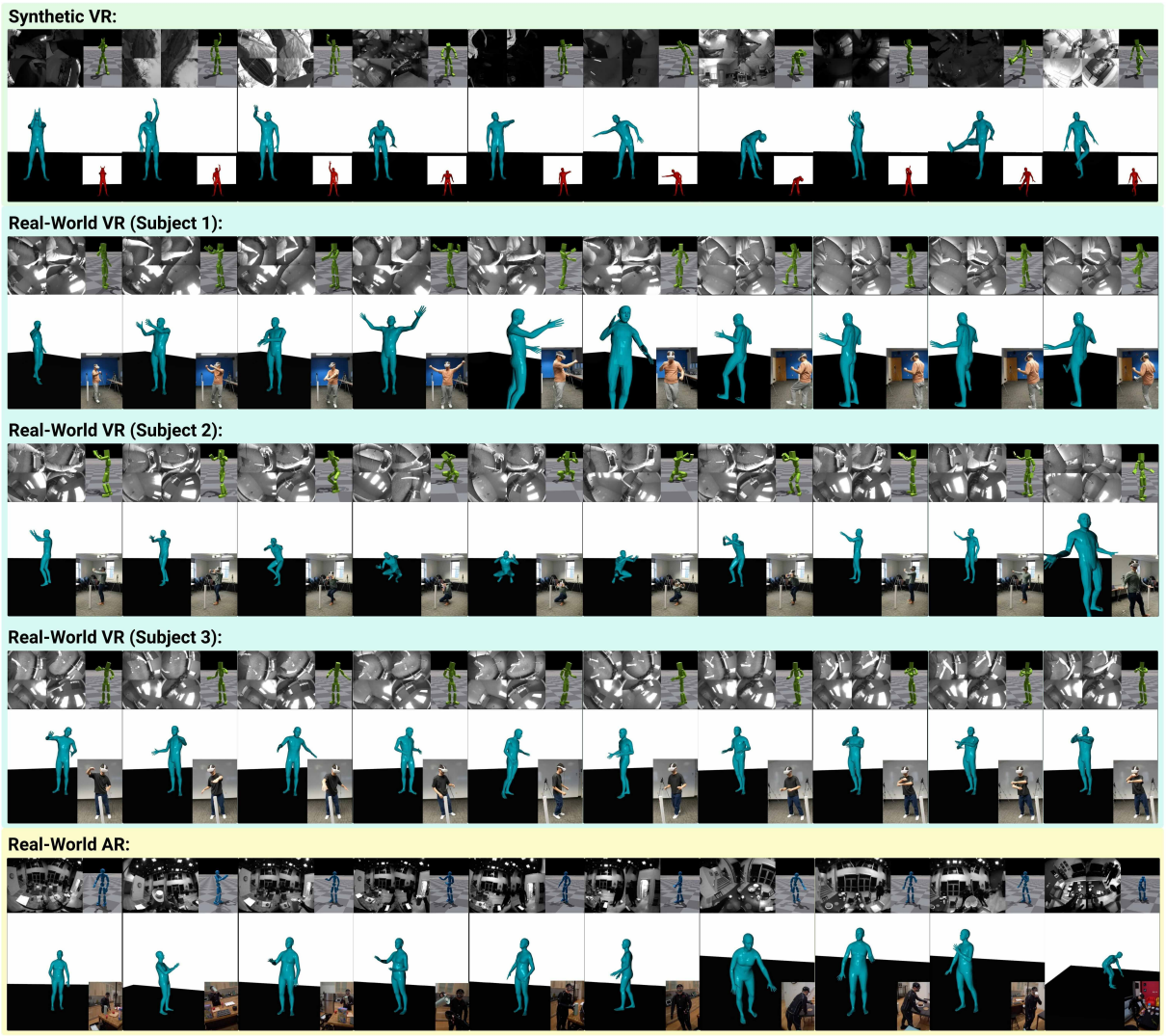


Figure 4.4. Qualitative results on synthetic and real-world AR/VR headset data. We visualize camera images, simulation, rendered mesh from simulation states, and third-person reference views. We show that our method can transfer to real-world data and handle diverse body poses including kicking, kneeling, *etc*. For AR headset results, the third-person view is provided by another subject wearing a headset.

our real-world dataset, where the scale and global position of the pseudo-ground-truth pose are noisy. To maintain the consistency of evaluation between the two humanoids, we select 11 common joints (head, left and right shoulders, elbows, wrists, knees, ankles) to report joint errors, rather than evaluating all joints as in the prior art [169]. To test physical realism, we include acceleration E_{acc} (mm/frame^2) and velocity E_{vel} (mm / frame) errors.

Baselines. We adopt the SOTA vision-based pose estimation method UnrealEgo [7] as the main vision-based baseline. UnrealEgo uses a U-Net structure to first reconstruct 2D heatmaps from input images. Then, an autoencoder is used to lift the 2D heatmap to 3D keypoints. UnrealEgo predicts the 3D pose in the headset's coordinate system instead of the global one. To compare against a physics-based method, we reimplement KinPoly [172] and add image input to create KinPoly-v. We also remove its dependence on action labels and external forces (replacing UHC [172] with PHC [169]). KinPoly-v uses a two-stage

	ADT-Test					
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$	E_{acc} ↓	E_{vel} ↓
KinPoly [172]	98.3%	93.8	80.6	60.8	5.9	9.5
UnrealEgo [7]	-	-	131.5	71.5	26.7	20.4
Ours (headset-only)	**100%**	120.7	120.6	85.8	5.2	9.2
Ours	**100%**	**67.8**	**67.6**	**47.7**	**4.6**	**7.3**

Table 4.3. Pose estimation results on the ADT test set (25k frames).

method: first estimate full-body pose from images, and then feed them into a pretrained imitator to drive the simulated avatar. It uses a closed-loop system, where the simulated state is fed into the image-based pose estimator, making it "dynamically regulated". KinPoly-v shares the same overall architecture as our method but uses kinematic pose as an intermediate representation to communicate with a pretrained imitator, instead of an end-to-end approach.

Implementation Details. We use NVIDIA's Isaac Gym [177] for physics simulation. All monochrome images are resized to 120×160 for the Aria and Quest headsets during training and evaluation. All MLPs used are 6-layer with units [2048, 1536, 1024, 1024, 512, 512] and SiLU [69] activation. The networks for Quest and Aria share the same architecture, with the only difference being the first layer for the image feature extractor (processing 2 or 4 monochrome images). Due to the orders-of-magnitude increase in input size and network capacity (ResNet 18 vs. 6 layer MLPs), training image-based methods with simulation is around 10 times more resource intensive even using our lightweight networks. We train π_{SimXR} for three days, collecting 0.1B environment interactions. For comparison, PHC trained for three days using RL collect around 2B environment interactions. After training, our estimation network and simulation run at ∼30 FPS. We perform all the evaluation with a fixed body shape for both humanoids, as our pipeline does not infer or use any body shape information. For more implementation details, please refer to the supplement.

4.4.1 Results

As motion is best seen in videos, please refer to our supplement for extended visual results of our proposed method.

	Synthetic-Test, $E_{\text{pa-mpjpe}}$ ↓										
Method	Head	L_Shoulder	L_Elbow	L_Wrist	R_Shoulder	R_Elbow	R_Wrist	L_Knee	L_Ankle	R_Knee	R_Ankle
UnrealEgo	**28.3**	29.3	40.7	55.8	29.2	38.2	46.9	53.0	74.4	50.9	72.8
Ours	30.2	**25.1**	**31.8**	**46.1**	**24.5**	**31.4**	**43.1**	**43.9**	**61.0**	**43.8**	**60.8**

Table 4.4. Per-joint error analysis on the $E_{\text{pa-mpjpe}}$ on the synthetic test set.

VR Headset Pose Estimation. In Table 4.2 and Fig. 4.4, we report the results of our synthetic and real-world dataset. Our method achieves comparable or better pose estimation results than both prior vision and vision + physics-based methods, estimating physically plausible full-body poses. Compared to UnrealEgo, we can see that SimXR uses fewer parameters by an order of magnitude while achieving a better pose estimation result. While both UnrealEgo and SimXR are single-frame methods (using no temporal architectures), SimXR has significantly less jittering. This is due to the laws-of-physics as a strong prior, and the inherent temporal information provided by the simulation state. Note that UnrealEgo es-

Synthetic-Test							
Vision	Headset	GN	Distill	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{acc} ↓	E_{vel} ↓
✗	✓	✓	✓	73.0%	118.9	8.8	11.6
✓	✗	✓	✓	27.1%	161.5	6.8	**8.3**
✓	✓	✗	✓	93.8%	74.9	7.3	9.3
✓	✓	✓	✗	35.8%	226.1	9.6	9.9
✓	✓	✓	✓	**94.3%**	**66.4**	**6.5**	**8.3**

Table 4.5. Ablations on components of PULSE: without vision/headset input, not suing group norm, and no distillation.

timates pose in the device frame, so it provides a better headset-relative pose estimate in terms of E_{mpjpe}. Our method directly estimates pose in the global coordinate system and does not benefit from aligning the head position (as can be seen in the small gap between global $E_{\text{g-mpjpe}}$ and local E_{mpjpe} joint errors. Compared to KinPoly-v, we can see that our method achieves better performance across the board. In KinPoly-v, the imitator, UHC [172], uses an external non-physical force to help balance the humanoid and does not require any reference velocity $\hat{\dot{q}}_t$ as input. PHC does not use any non-physical forces and does require reference velocities as input, which creates additional difficulty in KinPoly-v. As a result, KinPoly-v does not perform well in both synthetic and real-world test sets, showing the limitations of the two-stage methods. Open-loop methods (where the policy does not have access to the kinematic state) may suffer less from the velocity estimation issue but introduce more disjoint between the kinematic and dynamic processes, as shown in KinPoly [172]. The relatively small gap between real-world and synthetic performance shows that our synthetic dataset is effective in providing a starting point for this challenging task.

AR Headset Pose Estimation. Table 4.3 shows the test result on the ADT dataset. ADT is a much simpler dataset in terms of motion complexity, as it contains only walking, reaching, and daily activities. However, estimating pose from an AR headset is a harder task, as the viewing angle is much more challenging with the body not seen most of the time. Thus, UnrealEgo performs poorly on the dataset, unable to deal with unseen body parts. Our method can effectively leverage head movement and create plausible full-body motion. Due to similar issues in the VR headset case, KinPoly-v also does not perform well. To show that our method effectively uses the image input, Table 4.3 also includes a headset-only variant of our approach, where we do not use any vision-based input. Comparing row 3 (R3) and R4, we can see that vision-based input indeed provides valuable information about the movement of the hands.

4.4.2 Ablations and Analysis

In Table 4.5, we ablate the components of our method on the synthetic test set. Comparing R1, R2 and R5, we can see the importance of each of the two modalities: the vision signals provide most of the end-effector body movement signals, while the headset guides the body root motion. Without vision signals, the humanoid would achieve poor pose estimation results but can still achieve a reasonable success rate since the headset pose provides a decent amount of movement signals (R1). Without headset pose signals (R2), the humanoid will soon lose track of the head pose. In this case, the method needs to perform both SLAM and pose estimation from images, which requires special architectures. Comparing R3 and R5, we can see that the use of group normalization instead of batch normalization provides some boost in performance. R4 shows that training from scratch using RL for vision-based methods without using distillation would require more efficient algorithms.

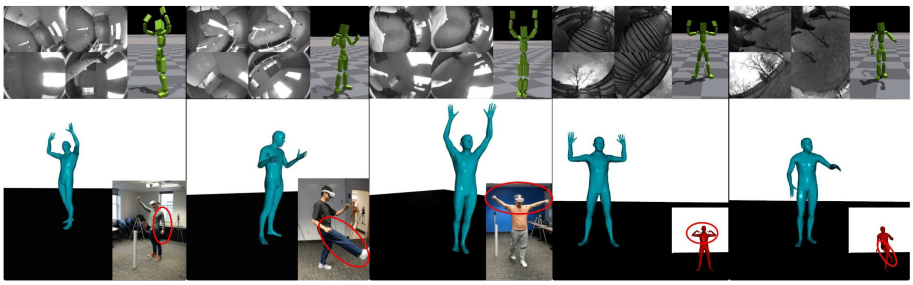


Figure 4.5. Failure cases of our method: misplaced feet or hands.

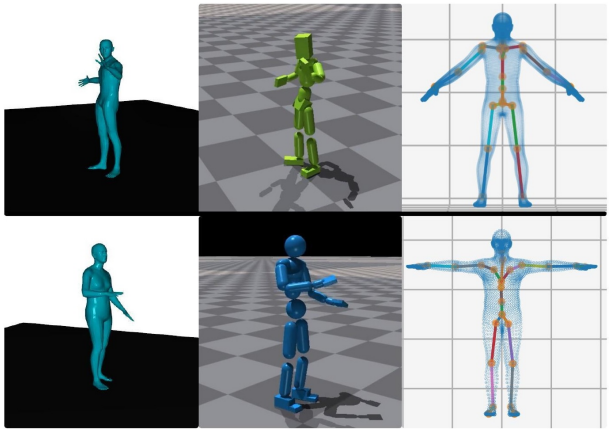


Figure 4.6. The two human models we use, their rendered mesh, simulated humanoid, and kinemaitc structure. (*Top*): our in-house humanoid with 24 DOF. (*Bottom*): SMPL humanoid with 23 DOF.

To further analyze our pose estimation results, we report the per-joint values for $E_{\text{pa-mpjpe}}$ in Table 4.4. We can see that SimXR, similar to UnrealEgo, makes the most mistakes in end effectors such as the wrists and ankles. It performs worse in head alignment since it uses a humanoid to track the headset pose, but gains performance on lower-body joints such as ankles. This is due to the fact that SimXR effective uses physics as a prior and can create plausible lower body movement based on input signals.

4.5 Additional Details

4.5.1 Supplementary Site & Videos

In the supplement site, we provide an extensive qualitative evaluation of our method. We visualized all three subjects and **full** sequences of our real-world Quest 2 data capture, as well as results from the synthetic dataset. From the videos, we can see that our end-to-end method can follow the headset wearer's body motion closely in a physically plausible fashion. We also show results on AR/Aria glasses and compare with SOTA vision-based and physics-based methods. Last but not least, we visualize failure cases of our method.

Figure 4.7. Self-occlusion visualization; blue dots are keypoints.

4.5.2 Implementation Details

4.5.3 Synthetic Data Generation

Humanoid Kinematic Structure. To create the synthetic egocentric data, we use an internal human mesh model similar to SMPL [161]. Given kinematic body rotations and scale parameters, we can create its corresponding mesh as shown in Fig. 4.6. We use a process similar to that of the SMPL humanoid to create an IsaacGym compatible humanoid for the in-house human mesh model.

MoCap Dataset. To generate synthetic data, we use a large-scale internal MoCap dataset consisting of 130 subjects and > 1300 capture sessions. The motion capture dataset contains a large number of daily activities (walking, running, gesturing, yoga, dancing, balancing, sitting, interaction with objects *etc.*). We remove sequences that contain human-object interactions that are not possible to mimic without simulating the objects (such as sitting on chairs). Since each capture session contains a long sequence of motion, we further divide the sessions into sequences that contain around $\sim$450 frames of motion, resulting in a total of 5169 sequences for training and testing.

Rendering Pipeline. As mentioned in the main paper, rendering is done using the exact placement, intrinsic, and distortion of the cameras as the Quest 2 headset. The Unity [21] game engine is used for rendering. In Fig. 4.8, we show examples of raw RGB images rendered using our pipeline. In each frame, we randomize the clothing, lighting, and background of the subjects as domain randomization. The background is rendered using a random image projected onto a skybox. We render each frame of motion from the MoCap dataset in 30 FPS. Each RGB image is rendered in a $640 \times 480 \times 3$ resolution, and we convert the images to monochrome and shrink them to $160 \times 120 \times 1$ for training and testing SimXR.

	Synthetic-Test, $E_{\text{pa-mpjpe}} \downarrow$					
	Head	L_Shoulder	L_Elbow	L_Wrist	R_Shoulder	R_Elbow
In-frame %	0.0%	4.0%	61.3 %	92.4 %	18.1 %	75.8%
In-frame / Out-frame	- / 30.2	28.0 / **25.0**	**30.7** / 33.2	**42.7** / 86.3	25.6 / **24.3**	**30.4** / 34.3
	R_Wrist	L_Knee	L_Ankle	R_Knee	R_Ankle	
In-frame %	96.5 %	99.0 %	98.4 %	99.4%	98.9%	
In-frame / Out-frame	**41.1** / 94.2	43.9 / **39.9**	**60.4** / 74.4	**43.7** / 45.5	**60.3** / 72.7	

Table 4.6. Error analysis based on joint in-frame status. "In-frame" is not equivalent to "visible" due to self-occlusion.

Visibility Analysis. Here we conduct a visibility analysis on our generated synthetic data. As accounting for self-occlusion involves reprocessing the synthetic dataset and conducting ray-marching for each joint on the clothed mesh to ascertain visibility, we opt to use the "in-frame" (whether the joint is in one of the camera frames) statistics to approximate visibility. This measure is a reliable visibility indicator for upper body joints, but less so for the lower body, where self-occlusion and extreme viewpoints are more prevalent (see Figure 4.7). From Table 4.6 we can see that the shoulder and elbow joints are frequently

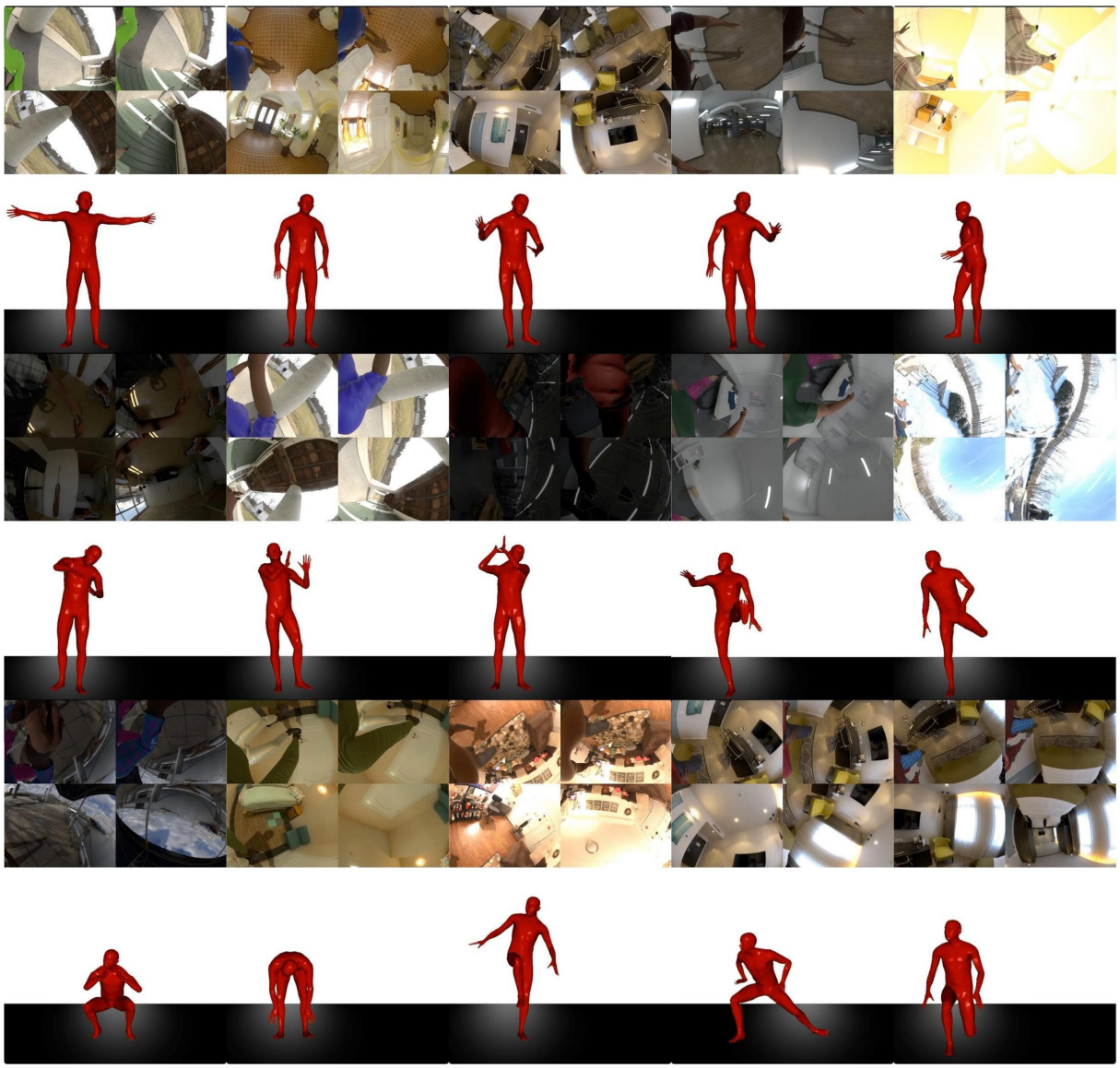


Figure 4.8. Sample synthetic data with various poses. Here we include the original rendered RGB images for demonstration purposes. We randomize the actor's clothes, background, lighting at every frame.

outside the frame, while the wrist and leg joints often remain within the frame. For the shoulder, elbow, and knee joints, their in-frame and out-frame results are similar, since they are closer to the torso. For the body extremities (wrists and ankles), there is a clear gap, as the largest errors occur out of the frame.

4.5.4 Details about SimXR

Body Shape Used for Evaluation. We conduct all our training and evaluation using a fixed body shape for both of our humanoids (SMPL and in-house). In other words, we use the mean body shape for SMPL and a fixed body shape for our internal humanoid and do not vary bone lengths between different motion sequences or subjects. Since our framework does not involve any intermediate representations such as 3D keypoints or poses, SimXR is scale invariant. When conducting real-world evaluations, we simply

	Batch Size	Learning Rate	# of samples	image size	Image-latent
SimXR	1024	5×10^{-4}	$\sim 10^8$	160×120	512
	Batch Size	Learning Rate	# of samples		
PHC	3072	2×10^{-5}	$\sim 10^{10}$		

Table 4.7. Hyperparameters for SimXR and PHC. Due to the increase in input size, SimXR is trained with significantly less samples than PHC and requires distillation.

	Synthetic-Train					
Method	Succ ↑	$E_{\text{g-mpjpe}} \downarrow$	$E_{\text{mpjpe}} \downarrow$	$E_{\text{pa-mpjpe}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
PHC	99.8%	25.6	20.4	15.5	2.4	3.5

Table 4.8. Motion imitation result by the pretrained imitator on the in-house MoCap dataset.

adjust the height of the headset pose to match the standing head positions of the mean body shape. This is done in a calibration phase in which the subject is standing still. This process is effective as SimXR can estimate the pose for the three subjects who have different heights. Notice that our imitator can be trained to handle different body shapes, but we opt out of this option as estimating body shape from the distorted egocentric views is still an unsolved problem.

Training Process. The training process for SimXR is similar to training a motion imitator, with the distinction being that we provide images and headset pose as input instead of full-body reference pose. To better learn harder motion sequences, we use the same hard-negative mining process proposed in PHC [169] and PULSE [170]. Concretely, during training, given the full motion and image dataset $\hat{\boldsymbol{Q}}$, we evaluate the current policy on the full dataset and pick the sequences that the policy fails to form $\hat{\boldsymbol{Q}}_{\text{hard}}$. We keep updating $\hat{\boldsymbol{Q}}_{\text{hard}}$ at intervals until the success rate no longer increases. The hyperparameters for training SimXR can be found in Table 4.7.

4.5.5 Details about Pretrained Imitators

For the SMPL humanoid (AR / Aria glass experiments), we use an off-the-shelf motion imitator, PHC [169], trained on the AMASS dataset. We do not make any additional modifications to PHC, since the motion in the ADT dataset is relatively simple. For the in-house humanoid and VR / Quest experiments, we train an imitator using the same training procedure and hyperparameters provided in the PHC implementation. We train the imitator using the training sequences from the internal MoCap dataset, achieving the imitation performance shown in Table 4.8. We can see that the imitator has a high success rate and a low joint error on the training data, which means that it is suitable to be used as a teacher for downstream tasks.

4.5.6 Details about KinPoly-v

We adapt KinPoly [172] to also consume images as input for egocentric pose estimation. In KinPoly, a policy is learned to produce kinematic full-body poses $\tilde{\boldsymbol{q}}_{t+1}$ based on the pose of the headset, which is then fed to an external force based motion imitator (UHC) for physics-based imitation. Comparing KinPoly to previous methods that use both an imitator and a pose estimator (*e.g.* SimPOE [329]), the main difference is whether the pose estimator is aware of the simulated humanoid state. In prior art, the pose estimator is not conditioned on the simulation state and operates independently from the physics

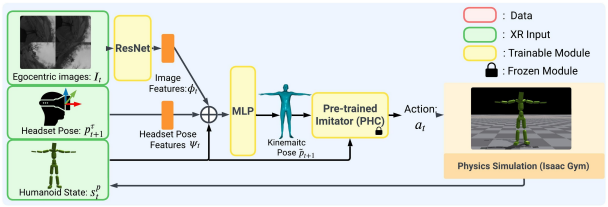


Figure 4.9. KinPoly-v's network architecture. Different from distilling from a pretrained imitator, KinPoly-v outputs kinematic pose to the pretrained imitator for physcis-based motion imtiation.

simulation. This creates an open-loop system where the pose estimator can estimate a pose that drifts far away from the simulation state, leading the imitator to fall. KinPoly aims to create a closed-loop system where the pose estimator also takes the simulation state into consideration. This methodology is also used in EmbodiedPose [173]. However, the main problem with KinPoly is that, while it is relatively easy to output the correct reference kinematic pose $\tilde{\boldsymbol{q}}_{t+1}$ for the current timestep, it is difficult to compute the correct velocities $\tilde{\dot{\boldsymbol{q}}}_{t+1}$. Due to the exceptional capabilities of the non-physical forces, UHC does not require reference velocities as input. Concretely, UHC's goal state is defined as $\boldsymbol{s}_t^{\text{g-mimic}} \triangleq (\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t)$, which does not contain any velocity information. As a result, KinPoly's kinematic policy only needs to predict body pose, but not velocity, which simplifies the learning problem. However, since PHC does not use any external forces [327], it requires accurate reference velocities as input.

To remove the dependency on external forces (change from UHC to PHC), we experimented with two forms of velocity prediction. The first approach is to compute the velocity as a finite difference between consecutive frames of the predicted reference poses: $\tilde{\dot{\boldsymbol{q}}}_{t+1} = \tilde{\boldsymbol{q}}_{t+1} - \tilde{\boldsymbol{q}}_t$. This formulation is problematic as large jumps in predicted poses can result in large velocities, which in turn lead to the imitator falling. Another approach is to directly predict the velocity as an output, which turns out to be more stable. We use this version as our implementation for KinPoly-v. However, this approach still suffers from inaccurate velocity prediction, as can be seen in the supplement videos: when the motion becomes faster and more dynamic (such as sports movement or jogging), it becomes difficult to predict the correct velocities for the motion imitator. This can also be a result of the image input, which can be noisy and detrimental to the network learning a good velocity prediction.

Network Architecture. KinPoly-v shares the same input and network architecture as SimXR, with the only distinction being the output: KinPoly-v outputs the kinematic pose $\tilde{\boldsymbol{q}}_{t+1}$ rather than the PD targets $\boldsymbol{a}_t$ for the joints. KinPoly-v's architecture can be found in Fig.4.9

4.5.7 Details about UnrealEgo

We use the official UnrealEgo implementation, and pick the ResNet18 version with imagenet initialization for a fair comparison with SimXR. To be compatible with monochrome images, we replace the CNN layer with a single-channel convolutional layer and keep the Siamese network structure. We follow the official implementation and first train the 2D heatmap estimation network. Then, using the frozen heatmap estimation network, we train a 3D pose estimator based on the 2D heatmap input. We train the networks for three days to convergence, using a similar compute budget as training SimXR.

4.5.8 Additional Ablations and Failure Cases

Recurrent Networks. Currently, SimXR is a per-frame model without using any temporal model architecture. SimXR does rely on temporal information in the form of a simulation state and estimates

	Synthetic-Test					
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	$E_{\text{pa-mpjpe}}$	E_{acc} ↓	E_{vel} ↓
Ours (GRU)	91.6%	78.2	73.7	52.8	8.8	10.4
Ours	**96.2%**	**69.0**	**65.2**	**43.2**	**6.8**	**8.7**

Table 4.9. Additional ablation on using recurrent architecture

temporally coherent motion by jointly considering humanoid state and input images. Based on the intuition that incorporating recurrent networks is essential to help robots complete tasks [348], we also tried recurrent architecture in our early experiments. We tested a simple GRU-based [49] architecture with 512 hidden units, and forms a lightweight Conv-LSTM [243]. Table 4.9 shows that the use of a recurrent network does not offer an immediate performance increase. We hypothesize that, for a pose estimation task with dense per-frame input, a recurrent network may not be necessary to ensure good performance. Further investigation is needed to better leverage the temporal coherence in videos.

Additional Failure Cases. In our supplementary videos, we visualize the common failure cases of SimXR. Being one of the first methods to drive simulated avatars from images and headset pose input from XR headsets, SimXR shows the feasibility of training such a network, but it is still far from perfect.

- We can observe that the humanoid stumbles and drags its feet to stay balanced when moving around, which is caused by ambiguity in movement signals. Since our humanoid has no information about the future movements of the camera wearer, it adopts the foot-dragging behavior to be cautious and stay balanced.
- Accurate foot movement can still be challenging: while SimXR can estimate kicking and raising feet, it can also miss raised feet due to the challenging viewing angle. The foot can be barely visible even when raised, and the color of the garment can create additional ambiguities.
- We can observe that when the hands are held perfectly still, the humanoid can have micro movements due to inaccuracy in inferring the body poses. Tackling this issue is challenging, as the movement of the headset can cause the hands to move in the camera space but not in the global space, and differentiating between the two requires further investigation.
- The humanoid can also have erroneous hands movement from time to time, erecting the hand quickly and putting them down due to image noise and occlusion.
- Another source of inaccuracy is fast and sporty movement, where the humanoid can lag behind in performing the actions or fall down.

In the future, our aim is to incorporate a larger MoCap and synthetic datasets to improve the robustness of the controller. Introducing auxiliary pose estimation losses during training could also improve SimXR.

Acknowledgement. We thank Zihui Lin for her help in making the plots in this paper. Zhengyi Luo is supported by the Meta AI Mentorship (AIM) program.

4.6 Conclusion and Future Work

Failure Cases. In Fig. 4.5, we visualize some failure cases of our method. As one of the first methods to control a simulated humanoid to match image observations from commercial headsets, it can misinterpret hand positions when they are not observed. Some kicking motion can also be ignored if the

feet observation is blurry (due to clothing color, lighting *etc.*). We also notice that, in some cases, the simulated humanoid movement might lag behind the real-world images, especially when hands come in and out of view. This can be attributed to the humanoid being conservative and not committing to movement until the body part is fully visible. In the videos, we can observe the humanoid stumbling and dragging its feet to remain balanced, especially when the headset moves too quickly.

Conclusions. We propose SimXR to control simulated avatars to match sensor input from commercially available XR headsets. We propose a simple, yet effective, end-to-end learning framework to learn to map from headset pose and camera input from XR headsets to humanoid control signals via distillation. To train our method and facilitate future research, we also propose a large-scale synthetic dataset for training and a real-world dataset for testing, all captured using standardized off-the-shelf hardware. Training only using synthetic data, our lightweight networks can control simulated avatars using real-world data capture with high accuracy in real-time. Future directions include adding auxiliary loss during training to improve the accuracy of pose estimation, incorporating temporal information, and using scene-level information for better pose estimates. While controlling simulated humanoids based on visual input takes an important step toward humanoid capabilities, it does not involve human-object interactions. The affinity of this task to motion imitation affords it the chance to be directly distilled from PHC, but tasks that involve humanoid-object interactions (tasks that PHC was not trained for) can not. In the next chapter, we form a universal humanoid control prior that facilitates skill reuse and speeds up downstream task training.

Part II

Forming Motor Skill Representations

Chapter 5

PULSE: Physics-based Universal Humanoid Motion Representation

5.1 Introduction

In the previous chapters, we leverage a pretrained motion imitator to solve pose estimation tasks. However, both approaches—kinematic policy and distillation—rely on ground-truth kinematic motion as supervision signals. Moreover, if the imitator is not trained with the ability to handle human-scene and human-object interactions, downstream tasks cannot acquire these capabilities without fine-tuning. In the following chapter, we propose an alternative approach that leverages PHC: learning a universal humanoid motion representation, allowing downstream RL tasks to directly utilize PHC's motor skills during training.

Due to advances in motion capture (MoCap) and reinforcement learning (RL), humanoids can now imitate entire datasets of human motion [169], perform dazzling animations [215], and track complex motion from sparse sensors [295]. However, each of these tasks requires the careful curation of motion data and training a new physics-based policy from scratch. This process can be time-consuming and engineering heavy, as any issue in reward design, dataset curation, or learning framework can lead to unnatural and jarring motion. Thus, though they can create physically realistic human motion, simulated humanoids remain inapplicable en masse.

One natural way to reuse pre-trained motion controllers is to form a latent space/skill embedding that can be reused in hierarchical RL. First, a task such as motion imitation [24, 183, 184, 299, 318] or adversarial learning [166, 213, 275] is used to form a representation space that can be translated into motor action using a trained decoder. Then, for new tasks, a high-level and task-specific policy is trained to generate latent codes as action, allowing efficient sampling from a structured action space and the reuse of previously learned motor skills. The main issue of this approach is the coverage of the learned latent space. Previous methods use small and specialized motion datasets and focus on specific styles of movements like locomotion, boxing, or dancing [93, 213, 275, 299, 318, 347]. This specialization yields latent spaces that can only produce a narrow range of behaviors preordained by the training data. Attempts to expand the training dataset, such as to CMU MoCap [53], have not yielded satisfactory results [299, 318]. Thus, these motion representations can only be applied to generative tasks such as locomotion and stylized animation. For tasks such as free-form motion tracking, they are limited by their motion latent spaces' coverage of the wide spectrum of possible human motion.

Another way to reuse motor skills is to formulate the low-level controller as a motion *imitator*, with the high-level control signal being full-body kinematic motion. This approach is frequently used in motion

tracking methods [169,172,173,297,328,329,331], and trained with supervised learning with paired input (*e.g.* video and kinematic motion). In generative tasks without paired data, RL is required. However, kinematic motion is a poor sampling space for RL due to its high dimension and lack of physical constraints (*e.g.* a small change in root position can lead to a large jump in motion). To apply this approach to generative tasks, an additional kinematic motion latent space such as MVAE [152,331] or HuMoR [231] is needed. Additionally, when agents must interact with their environment or other agents, using only kinematics as a motion signal does not provide insight into interaction dynamics. This makes tasks such as object manipulation and terrain traversal difficult.

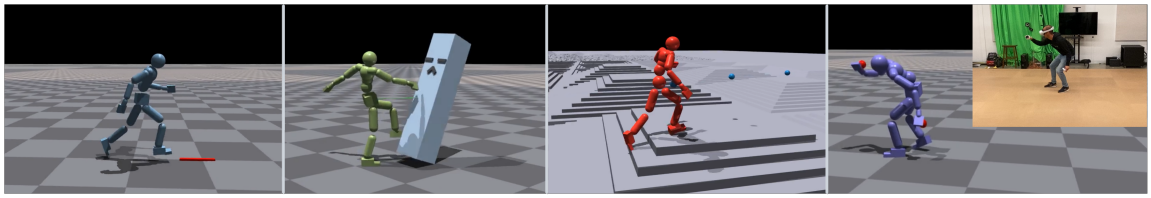


Figure 5.1. We propose to learn a motion representation that can be reused universally by downstream tasks. From left to right: speed, strike target, complex terrain traversal, and VR controller tracking.

In this work, we demonstrate the feasibility of learning a universal motion representation for humanoid control. Our proposed **P**hysics-based **U**niversal motion **L**atent **S**pac**E** (PULSE) is akin to a foundation model for control where downstream tasks ranging from simple locomotion, complex terrain traversal, to free-form motion tracking can all reuse this representation. To achieve this, we leverage recent advances in motion imitation [169] and distill comprehensive motor skills into a probabilistic latent space using a variational information bottleneck. This formulation ensures that the learned latent space inherits the motor skills of the motion imitator trained on a large-scale dataset. Our key insight lies in using a pretrained imitator for direct online distillation, which is crucial for effective scaling to large datasets.

To enhance our model's expressiveness, we jointly train a learnable prior conditioned on proprioception (humanoid's own pose and velocities). Random rollouts from this prior using Gaussian noise lead to diverse and stable humanoid motion. This is essential to improve sampling efficiency for downstream tasks since the high-level controller now samples coherent human motion for exploration instead of relying on applying noisy joint torques. The use of this prior in hierarchical RL speeds up training for the motion tracking task and leads to human-like behavior for generative tasks such as complex terrain traversal [233] without using an adversarial reward.

To summarize, our contributions are as follows: 1) We show that one can form a *universal* humanoid motion latent space by distilling from an imitator that can imitate *all* of the motion from a large scale motion dataset. 2) We show that random rollouts from such a latent space can lead to human-like motion. 3) We apply this motion representation to generative tasks ranging from simple locomotion to complex terrain traversal and show that it can speed up training and generate realistic human-like behavior when trained with simplistic rewards. When applied to motion tracking tasks such as VR controller tracking, our latent space can produce free-from human motion.

5.2 Related Work

Physics-based Humanoid Motion Latent Space. Recent advances often use adversarial learning or motion tracking objective to form reusable motion representation. Among them, ASE [213] and CALM [275] use a discriminator or encoder reward to encourage mapping between random noise and realistic behavior. Although effective on small and specialized datasets (such as locomotion and sword strikes), the formed latent space can fail to cover more diverse motor skills. Using explicit motion tracking, on the other hand, can potentially form a latent space that covers all the motor skills from the data. ControlVAE [318], NPMP [183], PhysicsVAE [299] and NCP [347] all follow this methodology. NCP learns the latent space jointly through the imitation task using RL, while ControlVAE and PhysicsVAE use an additionally learned world model. They show impressive results on tasks like point-goal, maze traversal, boxing, etc, but for tasks such as handling uneven terrains, adaptation layers are required [299]. We also use motion tracking as the task to learn motion representation, but form a *universal* latent space that covers 40 hours of MoCap data. We show that the key to a universal latent space is distilling from a pretrained imitator and a jointly learned prior.

Kinematics-based Human Motion Latent Space. Kinematics-based motion latent space has achieved great success in motion generation and pose estimation. In these approaches, motion representation is learned directly from data (without involving physics simulation) via supervised learning. Some methods obtain latent space by compressing multi-step motion into a single code [87, 119, 141, 164, 171, 216, 289, 326, 339]. Some, like MVAE [152] and HuMoR [231], are autoregressive models that model human motion at the per-frame level and are more applicable to interactive control tasks. While MVAE is primarily geared toward locomotion tasks, HuMoR is trained on AMASS [176] and can serve as a universal motion prior. Our formulation is close to HuMoR while involving control dynamics. Through random sampling, our latent space governed by the laws of physics can generate long and physically plausible motion, while HuMoR can often lead to implausible ones.

Physics-based Humanoid Motion tracking. From DeepMimic [208], RL-based motion tracking has gone from imitating single clips to large-scale datasets [46, 79, 169, 292]. Among them, a mixture of experts [297], differentiable simulation [234], and external forces [327] have been used to improve the quality of motion imitation. Recently, Perpetual Humanoid Controller (PHC) [169] allows a single policy to mimic almost all of AMASS and recover from falls. We improve PHC to track all AMASS and distill its motor skills into a latent space.

Knowledge Transfer and Policy Distillation. In Policy Distillation [241], a student policy is trained using teacher's collected experiences via supervised learning (SL). In kickstarting [244], the student collects its own experiences and is trained with both RL and SL. As we distill motor skills from a pretrained imitator, we also use the student for exploration and the teacher to annotate the collected experiences, similar to DAgger [239].

5.3 Preliminaries

We define the full-body human pose as $\boldsymbol{q}_t \triangleq (\boldsymbol{\theta}_t, \boldsymbol{p}_t)$, consisting of 3D joint rotation $\boldsymbol{\theta}_t \in \mathbb{R}^{J\times 6}$ and position $\boldsymbol{p}_t \in \mathbb{R}^{J\times 3}$ of all J links on the humanoid, using the 6 degree-of-freedom (DOF) rotation representation [346]. To describe the movement of human motion, we include velocities $\dot{\boldsymbol{q}}_{1:T}$, where $\dot{\boldsymbol{q}}_t \triangleq (\boldsymbol{\omega}_t, \boldsymbol{v}_t)$ consists of angular $\boldsymbol{\omega}_t \in \mathbb{R}^{J\times 3}$ and linear velocities $\boldsymbol{v}_t \in \mathbb{R}^{J\times 3}$. As a notation convention, we use $\widehat{\cdot}$ to denote the ground truth kinematic quantities from Motion Capture (MoCap) and normal symbols without accents for values from the physics simulation. A MoCap dataset is called $\hat{\boldsymbol{Q}}$. We use the terms "track", "mimic", and "imitate" interchangeably.

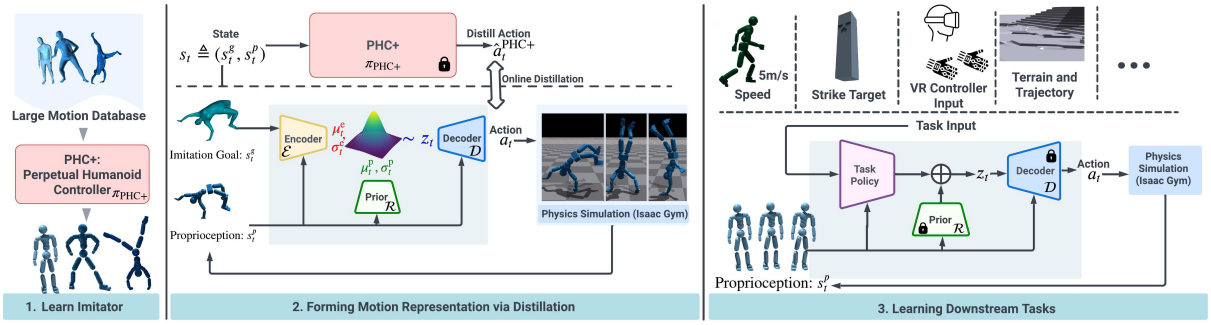


Figure 5.2. We form our latent space by directly distilling from a pretrained motion imitator that can imitate all of the motion sequences from a large-scale dataset. A variational information bottleneck is used to model the distribution of motor skills conditioned on proprioception. After training the latent space model, the decoder $\mathcal{D}_{\text{PULSE}}$ and prior $\mathcal{P}_{\text{PULSE}}$ are frozen and used for downstream tasks.

Goal-conditioned Reinforcement Learning for Humanoid Control. In this work, all tasks use the general framework of goal-conditioned RL. Namely, a goal-conditioned policy π is trained to complete tasks such as imitating the reference motion $\hat{\boldsymbol{q}}_{1:T}$, following 2D trajectories ${\boldsymbol{q}_t}^{\intercal}$, tracking 3D VR controller trajectories $\boldsymbol{p}^{\text{VR}}_{1:T}$, etc. The learning task is formulated as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$ of states, actions, transition dynamics, reward function, and discount factor. Physics simulation determines the state $\boldsymbol{s}_t \in \mathcal{S}$ and transition dynamics $\mathcal{T}$, where a policy computes the action $\boldsymbol{a}_t$. For humanoid control tasks, the state $\boldsymbol{s}_t$ contains the proprioception $\boldsymbol{s}^{\text{p}}_t$ and the goal state $\boldsymbol{s}^{\text{g}}_t$. Proprioception is defined as $\boldsymbol{s}^{\text{p}}_t \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t)$, which contains the 3D body pose $\boldsymbol{q}_t$ and velocity $\dot{\boldsymbol{q}}_t$. The goal state $\boldsymbol{s}^{\text{g}}_t$ is defined based on the current task. When computing the states $\boldsymbol{s}^{\text{g}}_t$ and $\boldsymbol{s}^{\text{p}}_t$, all values are normalized with respect to the humanoid heading (yaw). Based on the proprioception $\boldsymbol{s}^{\text{p}}_t$ and the goal state $\boldsymbol{s}^{\text{g}}_t$, we have a reward $r_t = \boldsymbol{\mathcal{R}}(\boldsymbol{s}^{\text{p}}_t, \boldsymbol{s}^{\text{g}}_t)$ for training the policy. We use proximal policy optimization (PPO) [245] to maximize discounted reward $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$. Our humanoid follows the kinematic structure of SMPL [161] using the mean shape. It has 24 joints, of which 23 are actuated, resulting in an action space of $\boldsymbol{a}_t \in \mathbb{R}^{23\times 3}$. Each degree of freedom is actuated by a proportional derivative (PD) controller, and the action $\boldsymbol{a}_t$ specifies the PD target.

5.4 Physics-based Universal Humanoid Motion Latent Space

To form the universal humanoid motion representation, we first learn to imitate all motion from a large-scale motion dataset (Sec.5.4.1). Then, we distill the motor skills of this imitator into a latent space model using a variational information bottleneck (5.4.2). Finally, after training this latent space model, we use it as an action space for downstream tasks (5.4.3). The pipeline is visualized in Fig.5.2.

5.4.1 PHC+: One Policy to Imitate Them All

For motion imitation, we follow the Perpetual Humanoid Controller (PHC [169]) and train a motion imitator that can recover from fallen states. $\pi_{\text{PHC}}(\boldsymbol{a}_t | \boldsymbol{s}^{\text{p}}_t, \boldsymbol{s}^{\text{g-mimic}}_t)$ is formulated as a per-frame motion imitation task, where at each frame, the goal state $\boldsymbol{s}^{\text{g-mimic}}_t$ is defined as the difference between proprioception and one frame reference pose $\hat{\boldsymbol{q}}_{t+1}$: $\boldsymbol{s}^{\text{g-mimic}}_t \triangleq \hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_{t+1} - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_{t+1}, \hat{\boldsymbol{p}}_{t+1})$. PHC utilizes a progressive training strategy, where a primitive policy $\boldsymbol{\mathcal{P}}^{(0)}$ is first trained on the entire motion dataset $\hat{\boldsymbol{Q}}$. The success rate is regularly evaluated on $\hat{\boldsymbol{Q}}$ and after it plateaus, sequences in which $\boldsymbol{\mathcal{P}}^{(0)}$ fails on form $\hat{\boldsymbol{Q}}^{(0)}_{\text{hard}}$, and a new primitive $\boldsymbol{\mathcal{P}}^{(1)}$ is initialized to learn $\hat{\boldsymbol{Q}}^{(0)}_{\text{hard}}$. This process continues

until $\hat{\boldsymbol{Q}}_{\text{hard}}^{(t)}$ cannot be learned or contains no motion. To learn to recover from the fail-state, an additional primitive $\boldsymbol{\mathcal{P}}^{(F)}$ is trained. After learning the primitives, a composer $\boldsymbol{\mathcal{C}}$ is trained to dynamically switch between frozen primitives. PHC achieves a 98.9% success rate on the AMASS training data, and we make small, yet critical modifications to reach 100% to form our imitator, PHC+.

First, extensive analysis of PHC reveals that the MoCap training dataset still contains sequences with severe penetration and discontinuity. These motions doubly impede PHC due to its hard-negative mining procedure, and removing them improves performance. Second, by immediately initializing $\boldsymbol{\mathcal{P}}^{(t+1)}$ to learn from $\hat{\boldsymbol{Q}}_{\text{hard}}^{(t)}$, PHC deprives $\boldsymbol{\mathcal{P}}^{(t)}$ of any opportunity to *further* improve using its own hard negatives. Focusing $\boldsymbol{\mathcal{P}}^{(t)}$ on $\hat{\boldsymbol{Q}}_{\text{hard}}^{(t)}$ before instantiating $\boldsymbol{\mathcal{P}}^{(t+1)}$ better utilizes $\boldsymbol{\mathcal{P}}^{(t)}$'s capacity, and coupled with some modernizing architectural changes, we use only three primitives to learn fail-state recovery and achieve a success rate of 100%.

5.4.2 Learning Motion Representation via Online Distillation

After learning PHC+ $\pi_{\text{PHC+}}(\boldsymbol{a}_t^{\text{PHC+}}|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ through RL, we learn a latent representation via online distillation. We adopt an encoder-decoder structure and use a conditional variational information bottleneck to model the distribution of motor skills $P(\boldsymbol{a}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ produced in $\pi_{\text{PHC+}}$. Although similar to the VAE architecture, our model does not use any reconstruction loss and is directly optimized in the action $\boldsymbol{a}_t$ space. Specifically, we have a variational encoder $\boldsymbol{\mathcal{E}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ that computes the latent code distribution based on current input states and a decoder $\boldsymbol{\mathcal{D}}_{\text{PULSE}}(\boldsymbol{a}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t)$ that produces action. Inspired by HuMoR [231], we employ a learned conditional prior $\boldsymbol{\mathcal{P}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}})$ instead of the zero-mean Gaussian prior used in VAEs. The learnable prior $\boldsymbol{\mathcal{P}}_{\text{PULSE}}$ allows the model to learn different distributions based on proprioception $\boldsymbol{s}_t^{\text{p}}$, as the action distribution for a person standing still and flipping midair can be significantly different. Formally, we model the encoder and prior distribution as diagonal Gaussian:

$$\boldsymbol{\mathcal{E}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}) = \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e), \boldsymbol{\mathcal{P}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}) = \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p). \tag{5.1}$$

During training, we sample latent codes from the encoder distribution $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e)$ and use the decoder $\boldsymbol{\mathcal{D}}_{\text{PULSE}}$ to compute the action. Combining the three networks, we obtain the student policy $\pi_{\text{PULSE}} \triangleq (\boldsymbol{\mathcal{E}}_{\text{PULSE}}, \boldsymbol{\mathcal{D}}_{\text{PULSE}}, \boldsymbol{\mathcal{P}}_{\text{PULSE}})$. Together with the learnable prior, the objective can be written as:

$$\log P(\boldsymbol{a}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}) \geq E_{\boldsymbol{\mathcal{E}}_{\text{PULSE}}}[\log \boldsymbol{\mathcal{D}}_{\text{PULSE}}(\boldsymbol{a}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t)] - D_{\text{KL}}(\boldsymbol{\mathcal{E}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})||\boldsymbol{\mathcal{P}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}})) \tag{5.2}$$

using the evidence lower bound. The data term $E_{\boldsymbol{\mathcal{E}}_{\text{PULSE}}}$ is similar to the reconstruction term in VAEs, and the KL term encourages the distribution of the latent code to be close to the learnable prior. To optimize this object, the loss function is written as:

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \alpha\mathcal{L}_{\text{regu}} + \beta\mathcal{L}_{\text{KL}}, \tag{5.3}$$

which contains the data term $\mathcal{L}_{\text{action}}$, the learnable prior $\mathcal{L}_{\text{KL}}$ from Eq.5.2, and a regulization term $\mathcal{L}_{\text{regu}}$. For online distillation, we have $\mathcal{L}_{\text{action}} = \|\boldsymbol{a}_t^{\text{PHC+}} - \boldsymbol{a}_t\|_2^2$, similar to the reconstruction term in VAEs. In our experiments, we find that if unconstrained, the learnable prior can push the latent codes far away from each other even for $\boldsymbol{s}_t^{\text{p}}$ that are close. Thus, we introduce an additional regularization term $\mathcal{L}_{\text{regu}} = \|\boldsymbol{\mu}_t^e - \boldsymbol{\mu}_{t-1}^e\|_2^2$ that penalizes a large deviation of consecutive latent codes. Intuitively, temporally close transitions should have similar latent representations, providing a smoother and more continuous latent space. $\mathcal{L}_{\text{regu}}$ serves as a weak prior, similar to the AR(1) prior used in NPMP [183]. In ablations, we show that using this prior is crucial for downstream tasks, without which the latent space can be very discontinuous and hard to navigate during RL exploration. To maintain the delicate balance of reconstruction error and KLD, we anneal β gradually.

For training π_{PULSE}, we need pairs of $(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}, \boldsymbol{a}_t, \boldsymbol{a}_t^{\text{PHC+}})$, which are obtained by rolling out π_{PULSE} in the environment for the motion imitation task and querying $\pi_{\text{PHC+}}$. While we can optimize π_{PULSE} with the RL objective by treating the decoder $\mathcal{D}_{\text{PULSE}}$ as a Gaussian distribution with fixed diagonal covariance matrix (similar to kickstarting [244]), we find that sampling the latent code together with random exploration sampling for RL introduces too much instability to the system. In our experiments, we show that training with RL objectives alone is not sufficient to form a good latent space, and optimizing with RL and supervised objectives combined creates a noisy latent space that is worse than without. The same as PHC+'s training process, we perform hard-negative mining and form $\hat{\boldsymbol{Q}}_{\text{hard}}$ for progressive training. When evaluating π_{PULSE}, we use the mean latent code $\boldsymbol{\mu}_t^e$ instead of sampling from $\mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e)$. This progressive training adds the benefit of balancing the data samples of the rarer and harder training motion sequences with the simpler ones.

5.4.3 Hierarchical Control for Downstream Tasks

After π_{PULSE} has converged, we can train a high-level policy $\pi_{\text{task}}(\boldsymbol{z}_t^{\text{omnigrasp}}|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ for downstream tasks. The frozen decoder $\mathcal{D}_{\text{PULSE}}$, together with the simulation, can now be treated as the new dynamcis system [89], where the action is now in the $\boldsymbol{z}_t$ space. Each high-level task policy $\pi_{\text{task}}(\boldsymbol{z}_t^{\text{omnigrasp}}|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}}) = \mathcal{N}(\boldsymbol{\mu}^{\text{task}}(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}}), \boldsymbol{\sigma}^{\text{task}})$ represents a Gaussian distribution with fixed diagonal covariance. We test on a suite of generative and motion tasks, and show that sampling from our prior, our policy can create realistic human-like behavior for downstream tasks, without relying on any adversarial learning objectives. The full training pipeline can be found in Alg.5.

Hierarchical Control with Learnable Prior. During high-level policy π_{task} training, it is essential to sample action from the prior distribution induced by $\mathcal{P}_{\text{PULSE}}$. Intuitively, if PULSE's latent space encapsulates a broad range of motor skills, randomly sampled codes may not lead to coherent motion. The learnable prior provides a good starting point for sampling human-like behavior. To achieve this, we form the action space of π_{task} as the residual action with respect to prior's mean $\boldsymbol{\mu}_t^p$ and compute the PD target ${\boldsymbol{a}_t}^{\text{task}}$ for downstream tasks as:

$$ {\boldsymbol{a}_t}^{\text{task}} = \mathcal{D}_{\text{PULSE}}(\pi_{\text{task}}(\boldsymbol{z}_t^{\text{omnigrasp}}|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}}) + \boldsymbol{\mu}_t^p), \tag{5.4} $$

where $\boldsymbol{\mu}_t^p$ is computed by the prior $\mathcal{P}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}})$. For RL exploration for π_{task}, we use a fixed variance (0.22) instead of $\boldsymbol{\sigma}_t^p$ as we notice that $\boldsymbol{\sigma}_t^p$ tends to be rather small. In ablations, we show that not using this residual action formulation for downstream task leads to a significant drop in performance.

5.5 Experiments

Tasks. We study a suite of popular downstream tasks used in the prior art. For generative tasks, we study reaching a certain x-directional speed (between 0 m/s and 5m/s), striking a target with the right hand (target initialized between 1.5∼5m), reaching a 3D point with the right hand (point initialized within 1 meters of the humanoid), and following a trajectory on complex terrains (slope, stairs, uneven surfaces, and obstacles). For motion tracking, we study a VR controller tracking task, where the humanoid tracks three 6DOF rigidbodies in space (two hand controllers and a headset). Note that the VR controller tracking task can be viewed as a generalization of the full-body motion imitation task, where only partial observation is provided. It is challenging since it requires the latent space to contain motor skills to match arbitrary user input in a continuous and streaming fashion.

Datasets. For training PHC+, PULSE, and VR controller policy, we use the cleaned AMASS training set. For strike and reach tasks, we use the AMASS training set for initial state sampling. For speed and terrain traversal tasks, we follow Pacer [233] to use a subset of AMASS that contains only locomotion for

Algo 5: Learn PULSE and Downstream Tasks

Function `TrainPULSE`($\mathcal{E}_{PULSE}, \mathcal{D}_{PULSE}, \mathcal{P}_{PULSE}, \hat{\mathcal{Q}}, \pi_{PHC+}$)**:**

  Input: Ground truth motion dataset $\hat{\mathcal{Q}}$, pretrained PHC+ $\pi_{\text{PHC+}}$, encoder $\mathcal{E}_{\text{PULSE}}$, decoder $\mathcal{D}_{\text{PULSE}}$, and prior $\mathcal{P}_{\text{PULSE}}$;

  while *not converged* **do**

    $\mathcal{M} \leftarrow \emptyset$ initialize sampling memory ;

    while $\mathcal{M}$ *not full* **do**

      $\hat{\boldsymbol{q}}_{1:T}, \boldsymbol{s}_t^{\text{p}} \leftarrow$ sample motion and initial state from $\hat{\mathcal{Q}}$;

      for $t \leftarrow 1...T$ **do**

        $\boldsymbol{s}_t \leftarrow \left(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}\right)$;

        $\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e \leftarrow \mathcal{E}_{\text{PULSE}}(\boldsymbol{z}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ `// encode latent;`

        $\boldsymbol{z}_t \sim \mathcal{N}(\boldsymbol{z}_t | \boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e)$ `// reparameterization trick for sampling latents;`

        $\boldsymbol{a}_t \leftarrow \mathcal{D}_{\text{PULSE}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t)$ `// decode action;`

        $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulation;`

        store $\boldsymbol{s}_t$ into memory $\mathcal{M}$;

    $\boldsymbol{a}_t^{\text{PHC+}} \leftarrow \pi_{\text{PHC+}}(\boldsymbol{a}_t^{\text{PHC+}} | \boldsymbol{s}_t)$ Annotate collected states in $\mathcal{M}$ using $\pi_{\text{PHC+}}$;

    $\boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p \leftarrow \mathcal{P}_{\text{PULSE}}(\boldsymbol{z}_t | \boldsymbol{s}_t^{\text{p}})$ Compute prior distribution based on proprioception ;

    $\mathcal{P}_{\text{PULSE}}, \mathcal{D}_{\text{PULSE}}, \mathcal{E}_{\text{PULSE}} \leftarrow$ supervised update for encoder, decoder, and prior using pairs of $(\boldsymbol{a}_t, \boldsymbol{a}_t^{\text{PHC+}}, \boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p, \boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e)$ and Eq.4.1.

  return $\mathcal{E}_{\text{PULSE}}, \mathcal{D}_{\text{PULSE}}, \mathcal{P}_{\text{PULSE}}$;

Function `TrainDownstreamTask`($\mathcal{D}_{PULSE}, \mathcal{P}_{PULSE}, \hat{\mathcal{Q}}, \pi_{task}$)**:**

  Input: Pretrained PULSE's decoder $\mathcal{D}_{\text{PULSE}}$ and prior $\mathcal{P}_{\text{PULSE}}$, task definition, and motion dataset for initial state sampling (*e.g.* $\hat{\mathcal{Q}}$) ;

  while *not converged* **do**

    $\mathcal{M} \leftarrow \emptyset$ initialize sampling memory ;

    while $\mathcal{M}$ *not full* **do**

      $\boldsymbol{s}_t^{\text{p}} \leftarrow$ sample initial state from $\hat{\mathcal{Q}}$;

      for $t \leftarrow 1...T$ **do**

        $\boldsymbol{z}_t^{\text{omnigrasp}} \sim \pi_{\text{task}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ `// usee pretrained latent space as action space;`

        $\boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p \leftarrow \mathcal{P}_{\text{PULSE}}(\boldsymbol{z}_t | \boldsymbol{s}_t^{\text{p}})$ `// compute prior latent code;`

        $\boldsymbol{a}_t \leftarrow \mathcal{D}_{\text{PULSE}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t^{\text{omnigrasp}} + \boldsymbol{\mu}_t^p)$ `// decode action using pretrained decoder;`

        $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulation;`

        $\boldsymbol{r}_t \leftarrow \mathcal{R}(\boldsymbol{s}_t)$ `// compute reward;`

        store $(\boldsymbol{s}_t, \boldsymbol{z}_t, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$ into memory $\mathcal{M}$;

    $\pi_{\text{task}} \leftarrow$ PPO update using experiences collected in $\mathcal{M}$;

  return π_{task} ;

initial state sampling. All methods are trained using the same initial-state sampling strategy. For testing imitation and VR controller tracking policy, we use the AMASS test set, as well as a real-world dataset (14 sequences, 16 minutes) from QuestSim [295] collected using the Quest 2 headset.

Baselines. To compare with state-of-the-art latent space models, we train ASE [213] and CALM [275] with the officially released code. We train all the latent space models from scratch using the AMASS training set with around 1 billion samples. Then, we apply the learned latent space to downstream tasks. Both CALM and ASE use a 64-d latent code projected onto a unit sphere, and are designed to be multi-frame latent codes operating at 6 HZ. Since our policy produces a per-frame latent code (30 Hz), for a fair comparison, we also test against ASE and CALM operating at 30 Hz. Notice that by design,

	AMASS-Train*					AMASS-Test*				
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
PHC	98.9 %	37.5	26.9	3.3	4.9	97.1%	47.5	40.0	6.8	9.1
PHC+	**100 %**	**26.6**	**21.1**	**2.7**	**3.9**	**99.2%**	**36.1**	**24.1**	**6.2**	**8.1**
PULSE	99.8 %	39.2	35.0	3.1	5.2	97.1%	54.1	43.5	7.0	10.3

Table 5.1. Motion imitation result (*data cleaning) on AMASS train and test (11313 and 138 sequences).

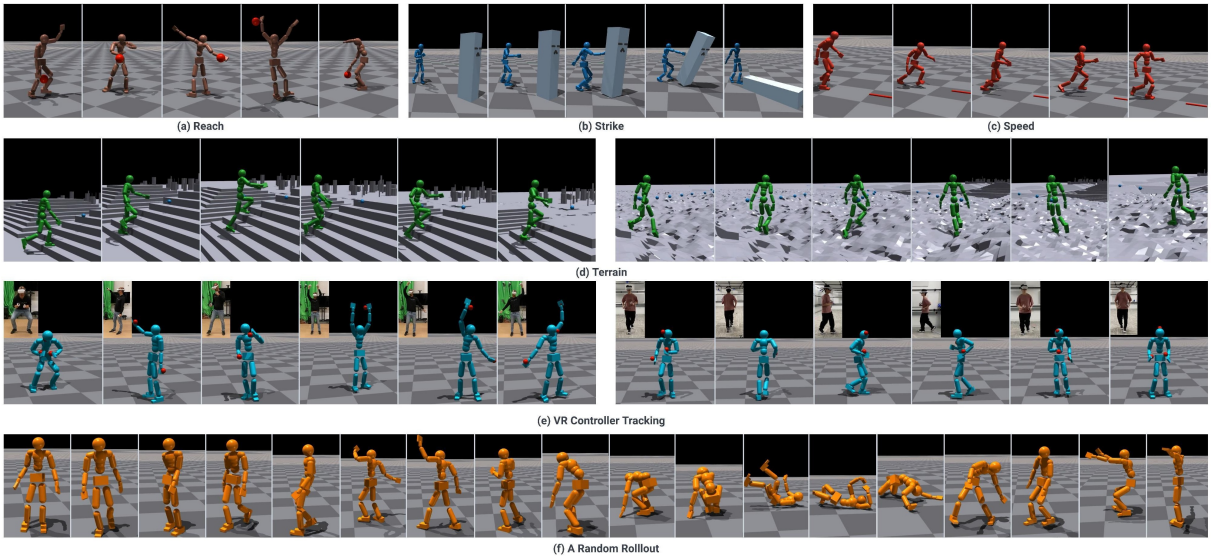


Figure 5.3. (a, b, c, d) Policy trained using our motion representation solves tasks with human-like behavior. (e) Our latent space is not constrained to certain movement styles and support free-form tracking. (f) Random sampling from learned prior $\mathcal{P}_{\text{PULSE}}$ leads to human-like movements as well as the recovery from fallen state.

CALM is trained to perform downstream tasks with user-specified skills. However, since we train our generative task without a specific style in mind, we opt out of the style reward and only train CALM with the task reward. For all tasks, we also compare against a policy trained from scratch without latent space or adversarial reward.

Metrics. For generative tasks (reach, speed, strike, and terrain), we show the training curve by visualizing the undiscounted return normalized by the maximum possible reward per episode. For motion imitation and VR controller tracking, we report the popular full body mean per joint position error for both global $E_{\text{g-mpjpe}}$ and root-relative E_{mpjpe} (in mm), and physics-based metrics such as acceleration error E_{acc} (mm/frame^2) and velocity error E_{vel} (mm/frame). Following PHC, we measure the success rate Succ (the success rate is defined as following reference motion < 0.5 m, averaged per joint, at all points in time). For VR controller tracking, the success rate is measured against only the three body points, but we report full-body $E_{\text{g-mpjpe}}$ and E_{mpjpe} when they are available (on AMASS). On the real-world dataset without paired full-body motion, we report controller tracking metrics $E_{\text{g-mhpe}}$ (mm) which computes the global position error of the three tracked controllers.

Implementation Details. Simulation is conducted at Isaac Gym [177], where the policy is run at 30 Hz and the simulation at 60 Hz. For PHC+ and all downstream tasks, each policy or primitive is a 6-layer MLP. Our encoder $\mathcal{E}_{\text{PULSE}}$, decoder $\mathcal{D}_{\text{PULSE}}$, prior $\mathcal{P}_{\text{PULSE}}$, CALM, and ASE are three-layer MLPs. The latent space is 32d: $z_t \in \mathbb{R}^{32}$, about half of the humanoid DOF of 69.

	AMASS-Train*					AMASS-Test*					Real-world			
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mhpe}}$ ↓	E_{acc} ↓	E_{vel} ↓
ASE-30Hz	79.8%	103.0	80.8	12.2	13.6	37.6%	120.5	83.0	19.5	20.9	7/14	99.0	28.2	20.3
ASE-6Hz	74.2%	113.9	84.9	81.9	57.7	33.3%	136.4	98.7	117.7	80.5	4/14	114.7	115.8	72.9
Calm-30Hz	16.6%	130.7	100.8	**2.7**	**7.9**	10.1%	122.4	99.4	**7.2**	**12.9**	2/14	206.9	**2.0**	**6.6**
Calm-6Hz	14.6%	137.6	103.2	58.9	47.7	10.9%	147.6	115.9	82.5	122.3	0/14	-	-	-
Scratch	98.8%	**51.7**	**47.9**	3.4	6.4	**93.4%**	**80.4**	**67.4**	8.0	13.3	**14/14**	**43.3**	4.3	6.5
Ours	**99.5%**	57.8	51.0	3.9	7.1	**93.4 %**	88.6	67.7	9.1	14.9	**14/14**	68.4	5.8	9.0

Table 5.2. Quantitative results on VR Controller tracking. We report result on AMASS and real-world dataset.

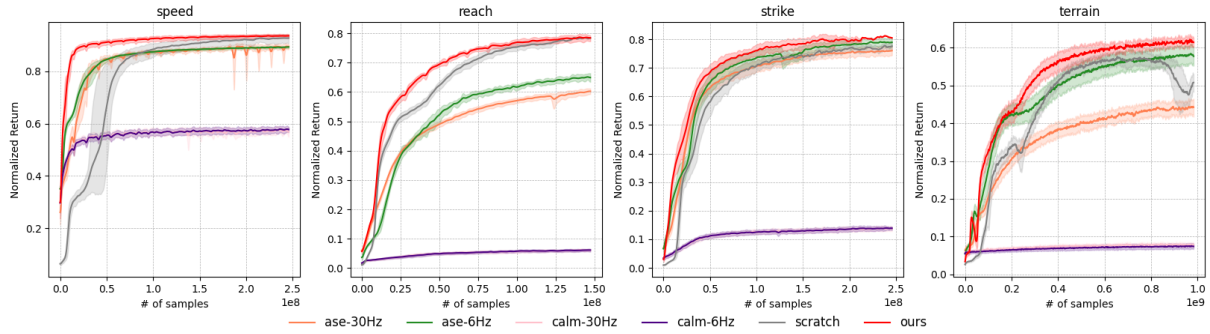


Figure 5.4. Training curves for each one of the generative tasks. Using our motion representation improves training speed and performance as the task policy can explore in an informative latent space. Experiments run for 3 times using different random seeds.

5.5.1 Motion Imitation

First, we measure the imitation quality of PHC+ and PULSE after distilling from PHC+. Table 5.1 shows that after distillation using the variational bottleneck, PULSE retains most of the motor skills generated by PHC+. This is crucial as this shows that the learned latent space model (the decoder $\mathcal{D}_{\text{PULSE}}$) inherits most of the motor skills to perform the majority of the AMASS dataset with high accuracy. Note that without the variational bottleneck, distillation *can reach* a 100% success rate, and the degradation in performance is similar to a non-zero reconstruction error of a VAE.

5.5.2 Downstream Tasks

Generative Tasks. We show the qualitative result in Fig.5.3 for the generative tasks (speed, reach, strike, and terrain). From Fig.5.4 we can see that using our latent space model as the low-level controller outperforms all baselines, including the two latent space models and training from scratch.

Compared to latent space models (ASE-30Hz, ASE-6Hz, CALM-30Hz, CALM-6Hz), our method consistently provides a better action space, achieving better normalized return across the board. This shows that the ASE and CALM formulation, though effective in learning motion styles from curated datasets, could not adequately capture motor skills in a large and unorganized dataset like AMASS. In general, 30 Hz models (ASE-30Hz and CALM-30Hz) outperform 6 Hz ones, signaling the need for finer-grained control latent space. In general, ASE outperforms CALM, partially due to CALM's motion encoder formulation. Without the style reward, exploration might be hindered for downstream tasks. For all tasks, training from scratch provides the closest normalized return to ours, but this could lead to unnatural behavior, as observed in ASE and our qualitative results. Our method also converges faster, even for the challenging terrain traversal task, where the humanoid needs to handle obstacles and stairs

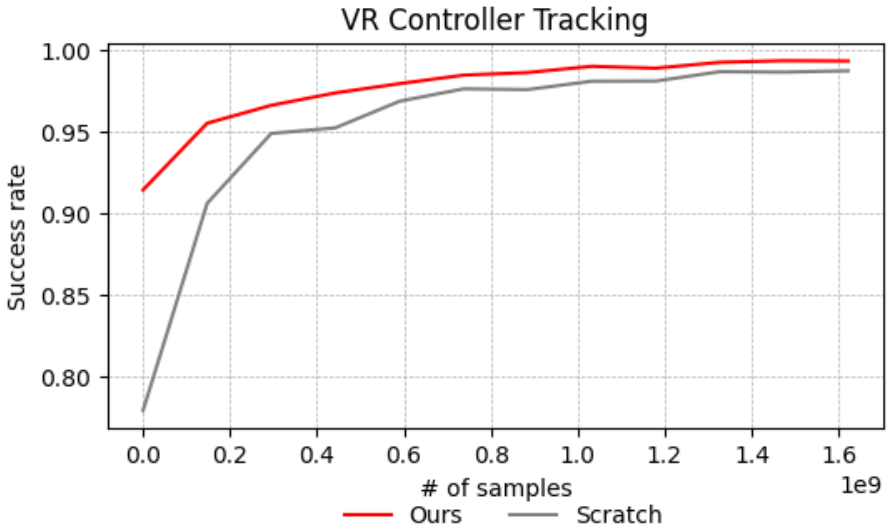


Figure 5.5. Success rate comparison between training from scratch and using our motion representation during training. Ours converges faster.

that it has not seen during training. Upon inspection, we can see that the humanoid uses human-like behavior for navigating the obstacles without using any style/adversarial reward as guidance. When stepping on stairs quickly, we can see that our humanoid employs motor skills similar to jumping. For detailed qualitative evaluation, see `supplement site`.

VR Controller Tracking. Unlike generative tasks, we can directly measure the success rate and joint position error for this task. In Table 5.2, we report the motion tracking result in the training, test, and real-world dataset. Continuing the trend of generative tasks, compared to other latent space models, ours significantly improves tracking error. This validates the hypothesis that PULSE can capture a wide range of human motor skills and can handle free-form motion tracking and pose estimation tasks. On the real-world dataset, our method can faithfully track all sequences. Note that our tracking result is comparable to training from scratch, which signifies that our latent space is diverse enough that it does not handicap exploration. Since the three-point tracking problem is relatively close to the full-body tracking problem, we expect training from scratch to perform better since it is a specialized tracking policy. Using our latent space also seems to be trading success rate for precise tracking, sometimes sacrificing precision in favor of staying stable. Figure 5.5 shows the training success rate between training from scratch and using our latent space. We can see that, using our latent space, the policy converges faster.

5.5.3 Random Motion Generation

As motion is best seen in videos, we show extensive qualitative random sampling and baseline comparison result in our `supplement site` and Fig.5.3 also shows a qualitative sample. Comparing with the kinematics-based, HuMoR, our method can generate realistic human motion without suffering from frequent implausible motion generation. Compared to other physics-based latent space, ASE and CALM, our representation has more coverage and can generate more natural and realistic motion. By varying the variance of the input noise, we can control the generation process and bias it toward a smoother or more energetic motion.

AMASS-Test									
idx	Prior	Prior-action	$\mathcal{L}_{\text{regu}}$	No RL	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
1	✗	✗	✗	✓	36.9%	114.6	79.4	8.1	15.8
2	✗	✗	✓	✓	45.6%	115.0	77.2	**7.5**	15.0
3	✓	✓	✗	✓	60.8%	106.8	72.4	9.4	16.0
4	✓	✓	✓	✗	71.0%	95.1	72.8	10.4	16.6
5	✓	✗	✓	✓	18.1%	108.6	83.8	11.2	16.4
6	✓	✓	✓	✓	**93.4%**	**88.6**	**67.7**	9.1	**14.9**

Table 5.3. Ablation on various strategies of learning the motion representation. We use the challenging VR controller tracking task to demonstrate the applicability of the latent space for downstream tasks. Prior: whether to use a learnable prior. Prior-action: whether to use the residual action with respect to the learned prior $\mathcal{P}_{\text{PULSE}}$. $\mathcal{L}_{\text{regu}}$: whether to apply $\mathcal{L}_{\text{regu}}$. No RL: whether to train PULSE together with the RL objective.

5.5.4 Ablations

In this section, we study the effects of different components of our framework using the challenging VR controller tracking task. During our experiments, we noticed that while the generative downstream tasks are affected less by the expressiveness of the latent space, the VR controller tracking task is. We train our latent space using the same teacher (PHC+) and train the VR controller tracking policy on the frozen decoder. When comparing Row 1 (R1) vs. R2 as well as R3 vs. R6, we can see that the regularization term $\mathcal{L}_{\text{regu}}$ that penalizes a large deviation between consecutive latent codes is effective in providing a more compact latent space for downstream exploration. Without such regularization, the encoder can create a discontinuous latent space that is harder to explore. R6 trains a policy that does not use the learnable prior but uses the zero-mean Gaussian $\mathcal{N}(0, 1)$ as in VAEs. R2 vs. R6 shows the importance of having a learnable prior, without which downstream sampling could be difficult. R5 vs. R6 shows that using the learnable prior as a residual action described in Eq.5.4 is essential for exploration, without which the latent space is too hard to sample from (similar to the issues with CALM without style reward). R4 vs. R6 shows that mixing the RL training and objective and online supervised distillation has a negative effect.

5.6 Additional Details

Extensive qualitative results are provided on the project page as well as in the supplementary zip (the zipped version is of lower video resolution to fit the upload size). As motion is best seen in videos, we highly encourage the readers to view them to better understand the capabilities of our method. Specifically, we evaluate motion imitation and fail-state recovery capabilities for PHC+ and PULSE after online distillation and show that PULSE can largely retain the abilities of PHC+. Then, we show long-formed motion generation result sampling from the PULSE' s prior and decoder. Sampling from PULSE, we can generate long-term, diverse and human-like motions, and we can vary the variance of the input noise to control the behaviors of random generation. We also compare with SOTA kinematics-based method, HuMoR, and show that while HuMoR can generate unnatural motions, ours are regulated by the laws of physics and remain plausible. Compared to SOTA physics-based latent space(ASE and CALM), our random generation appears more diverse. Finally, we show visualization for downstream tasks for both generative and estimation/tracking tasks and compare them with SOTA methods.

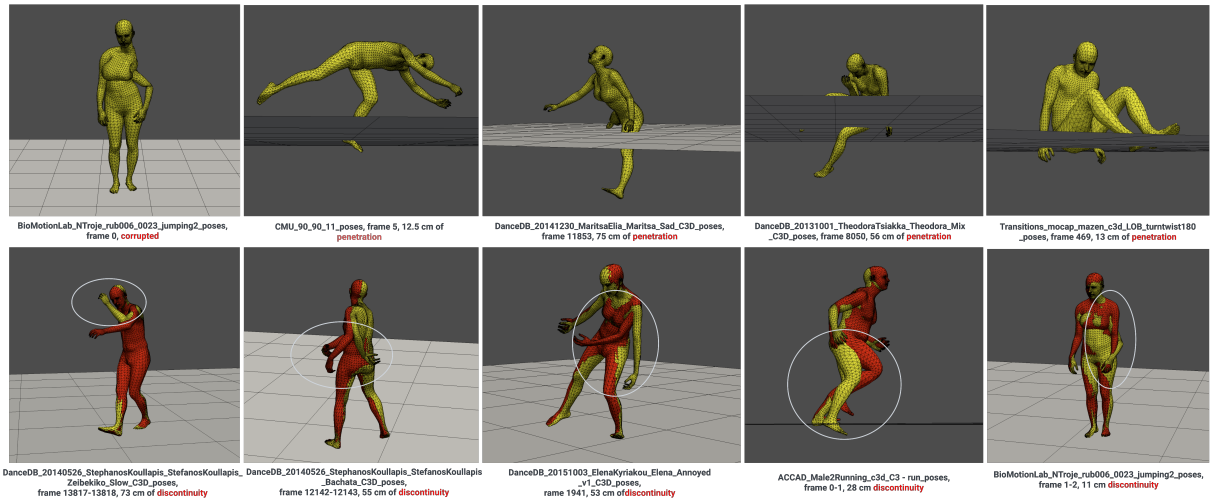


Figure 5.6. Visualization of issues in the AMASS dataset. Here we show sequences with corrupted poses, large penetration, and discontinuity. In the second row, the red and yellow mesh are 1 frame apart in 120Hz MoCap.

Method	Batch Size	Learning Rate	σ	γ	ϵ	w_{jp}	w_{jr}	w_{jv}	$w_{\text{j}\omega}$	# of samples
PHC+	3072	2×10^{-5}	0.05	0.99	0.2	0.5	0.3	0.1	0.1	$\sim 10^{10}$
	Batch Size	Learning Rate	α	β	Latent size	# of samples				
PULSE	3072	5×10^{-4}	0.005	$0.01 \rightarrow 0.001$	32	$\sim 10^{9}$				

Table 5.4. Hyperparameters for PHC+ and Pulse. σ: fixed variance for policy. γ: discount factor. ϵ: clip range for PPO. α: coefficient for $\mathcal{L}_{\text{regu}}$. β: coefficient for $\mathcal{L}_{\text{KL}}$.

5.6.1 Details about PHC+

5.6.2 Data Cleaning

We perform a failure case analysis and identified two main sources of imitation failure. First, we have dynamic motion, such as cartwheeling and consecutive back flips. Another, often overlooked, is that MoCap sequences can still have a large discontinuity and penetration due to failures in the MoCap optimization procedure or the fitting process [162]. After filtering out human-object interaction data following UHC [172], we found additional corrupted sequences in PHC's training data that have a large discontinuity or penetration. In Fig.5.6, we visualize some of the sequences we have identified and removed from the training data. Since we use the random state initialization proposed by DeepMimic [208], sampling frames that have large penetration could lead to the humanoid "flying off" from the ground as the physics simulation applies a large ground reactionary force. Naively adjusting the height of the sequence based on penetration could lead to floating sequences or discontinuity. Frames that have large discontinuities could lead to imitation failure or humanoid learning bad behavior to anticipate large jumps between frames. We remove these sequences and obtain 11313 training and 138 testing motion suitable for motion imitation training and testing, which we will release with the code and models.

5.6.3 Action and Rewards

Our action space, state, and rewards follow the specifications of the PHC paper. Specifically, the action $\boldsymbol{a}_t$ specifies the target for the proportional derivative (PD) controller on each of the 69 actuators. The

target joint is set to $\boldsymbol{q}_t^d = \boldsymbol{a}_t$ and the torque applied at each joint is $\boldsymbol{\tau}^i = \boldsymbol{k}^p \circ (\boldsymbol{a}_t - \boldsymbol{q}_t) - \boldsymbol{k}^d \circ \dot{\boldsymbol{q}}_t$. Joint torques are capped at 500 N-m. For the motion tracking reward, we use:

$$\begin{aligned} r_t &= 0.5 r_t^{\text{g-imitation}} + 0.5 r_t^{\text{amp}} + r_t^{\text{energy}}, \\ r_t^{\text{g-imitation}} &= w_{\text{jp}} e^{-100\|\hat{\boldsymbol{p}}_t - \boldsymbol{p}_t\|} + w_{\text{jr}} e^{-10\|\hat{\boldsymbol{q}}_t \ominus \boldsymbol{q}_t\|} + w_{\text{jv}} e^{-0.1\|\hat{\boldsymbol{v}}_t - \boldsymbol{v}_t\|} + w_{\text{j}\omega} e^{-0.1\|\hat{\boldsymbol{\omega}}_t - \boldsymbol{\omega}_t\|}, \end{aligned} \tag{5.5}$$

where $r_t^{\text{g-imitation}}$ is the motion imitation reward, r_t^{amp} is the discriminator reward, and r_t^{energy} an energy penalty. $r_t^{\text{g-imitation}}$ measures the difference between the translation, rotation, linear velocity, and angular velocity of the 23 rigid bodies in the humanoid. r_t^{amp}. r_t^{amp} is the Adversarial Motion Prior (AMP) reward [215], provided by a discriminator trained on the AMASS dataset. The energy penalty r_t^{energy} is $-0.0005 \cdot \sum_{j \in \text{ joints}} |\boldsymbol{\mu}_j \boldsymbol{\omega}_j|^2$ where $\boldsymbol{\mu}_j$ and $\boldsymbol{\omega}_j$ correspond to the joint torque and the joint angular velocity, respectively. The energy penalty [76] regulates the policy and prevents high-frequency jitter.

5.6.4 Model Architecture and Ablations

All primitives and composers in PHC + are a 6 layer MLP with units [2048, 1536, 1024, 1024, 512, 512] and SiLU activation. We find that changing the activation from ReLU [78, 191] to SiLU [101] provides a non-trivial boost in tracking performance. Combining with larger networks (from 3 layer MLP to 6 layer), we use only three primitives to learn fail-state recovery and achieve a success rate of 100%. To study the effect of the new activation function and the progressive training procedure, we perform ablation studies on the training of a single primitive $\boldsymbol{\mathcal{P}}$ (not the full PHC + policy) using the proposed changes.

Each primitive is trained for 3×10^9 samples. From Table 5.5, we can see that comparing Row (R1) and R3, the new progressive training procedure improves the success rate by a large amount, showing that $\boldsymbol{\mathcal{P}}$'s capacity is not fully utilized if $\hat{\boldsymbol{Q}}_{\text{hard}}$ is not formed and updated during each $\boldsymbol{\mathcal{P}}$'s training. When comparing R2 and R3, we can see that changing the activation function from ReLU to SiLU improves the tracking performance and improves E_{mpjpe}. Table.5.4 reports the hyperparameters we used for training.

		AMASS-Test				
Activation	Progressive	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
SiLU	✗	92.0%	43.0	29.2	6.7	8.9
ReLU	✓	97.8%	44.4	32.8	6.9	9.1
SiLU	✓	**98.5%**	**39.0**	**28.1**	**6.7**	**8.5**

Table 5.5. Ablations on PHC+'s primitive $\boldsymbol{\mathcal{P}}$ training. Progressive: refers to whether $\hat{\boldsymbol{Q}}_{\text{hard}}$ is updated during the primitive training (rather than waiting until convergence and initialize a new primitive).

5.6.5 Details about PULSE

5.6.6 Training Procedure

We train π_{PULSE} using the training procedure we used to train a primitive $\boldsymbol{\mathcal{P}}^{(0)}$ in PHC+, where we progressively form $\hat{\boldsymbol{Q}}_{\text{hard}}$ while training the policy. Since π_{PULSE} and $\pi_{\text{PHC+}}$ share the same state and action space, we query $\pi_{\text{PHC+}}$ at training time to perform online distillation. We anneal the coefficient β of $\mathcal{L}_{\text{KL}}$ from 0.01 to 0.001 starting from 2.5×10^9 to 5×10^9 samples. Afterward, β remains the same. We report our hyperparameters for training π_{PULSE} in Table. 5.4.

5.6.7 Comparison to Training Scratch without Distillation

One of our main contributions for PULSE is the using online distillation to learn π_{PULSE}, where the latent space uses knowledge distilled from a trained imitator, PHC+. While prior work like MCP [212] demonstrated the possibility of training such a policy from scratch (using RL without distillation), we

	AMASS-Train*					AMASS-Test*				
Distill	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
✗	72.0%	76.7	52.8	3.5	8.0	32.6%	98.4	79.4	9.9	16.2
✓	99.8 %	39.2	35.0	3.1	5.2	97.1%	54.1	43.5	7.0	10.3

Table 5.6. Ablations on training PULSE from scratch using RL (no distillation).

find that using the variational information bottleneck together with the imitation objective creates instability during training. We hypothesize that random sampling for the variational bottleneck together with random sampling for RL leads to noisy gradients. In Table 5.6, we report the result of motion imitation from training from scratch. We can see that the training using RL does not converge to a good imitation policy after training for more than 1×10^{10} samples.

5.6.8 Comparison to Other Latent Formulation (VQ-VAE, Spherical)

In our earlier experiments, we studied other forms of latent space such as a spherical latent space, similar to ASE [213], or a vector quantized latent space, similar to NCP [347]. For spherical embedding, we use the same encoder-decoder structure as in PULSE and use a 32-dimensional latent space normalized to the unit sphere. For a vector-quantized motion representation, we follow [154, 283] and use a 64-dimensional latent space divided into 8 partitions, using a dictionary size of 64. Dividing the latent space into partitions increases the representation power combinatorially [154] and is more effective than a larger dictionary size. Although through distillation, each of these representations could reach a high imitation success rate and MPJPE (spherical: 100% Succ and 28.1 $E_{\text{g-mpjpe}}$, VQ: 99.8 % Succ and 36.5 $E_{\text{g-mpjpe}}$), both lose the ability to serve as a generative model: random samples from the latent space do not generate coherent motion. In NPC, an additional prior needs to be learned. The quantized latent space also introduces artifacts, such as high-frequency jitter, since the network is switching between discrete codes. We visualize this artifact in our `supplement site`'s last section.

5.6.9 Downstream Tasks

Each generative downstream task policy π_{task} is a three-layer MLP with units [2048, 1024, 512]. For VR controller tracking, we use a six-layer MLP of units [2048, 1536, 1024, 1024, 512, 512]. The value function has the same architecture as the policy. All tasks are optimized using PPO. For simpler tasks (speed, reach, strike), we train for $\sim 2 \times 10^9$ samples. For the complex terrain traversal task, the policy converges after $\sim 1 \times 10^{10}$ samples. The strike, speed, and reach tasks follow the definition in ASE [213], while the following trajectory task follows PACER [233]. VR controller tracking task follows QuestSim [295].

Speed. For training the x-direction speed task, the random speed target is sampled between 0 m/s $\sim$ 5m/s (the maximum target speed for running in AMASS is around 5m/s). The goal state is defined as $\boldsymbol{s}_t^{\text{g-speed}} \triangleq (d_t, v_t)$) where d_t is the target direction and v_t is the linear velocity the policy should achieve at timestep t. The reward is defined as $r_{\text{speed}} = \text{abs}(v_t - \boldsymbol{v}_t{}^0)$ where $\boldsymbol{v}_t{}^0$ is the humanoid's root velocity.

Strike. For strike, since we do not have a sword, we substitute it with "strike with hands". The objective is to knock over the target object and is terminated if any body part other than the right hand makes contact with the target. The goal state $\boldsymbol{s}_t^{\text{g-strike}} \triangleq (\boldsymbol{x}_t, \dot{\boldsymbol{x}}_t)$) contains the position and orientation $\boldsymbol{x}_t$ as well as the linear and angular velocity $\dot{\boldsymbol{x}}_t$) of the target object in the agent frame. The reward is $r_{\text{strike}} = 1 - \boldsymbol{u}^{\text{up}} \cdot \boldsymbol{u}_t$ where $\boldsymbol{u}^{\text{up}}$ is the global up vector and $\boldsymbol{u}_t$ is the target's up vector.

Reach. For the reach task, a 3D point $\boldsymbol{c}_t$ is sampled from a 2-meter box centered at $(0, 0, 1)$, and the goal state is $\boldsymbol{s}_t^{\text{g-reach}} \triangleq (\boldsymbol{c}_t)$). The reward for reaching is the difference between the humanoid's right hand and

the desired point position $r_{\text{reach}} = \exp(-5\|\boldsymbol{p}_t^{\text{right hand}} - \boldsymbol{c}_t\|_2^2)$.

Trajectory Following on Complex Terrains . The humanoid trajectory following on complex terrain task, used in PACER [233], involves controlling a humanoid to follow random trajectories through stairs, slopes, uneven surfaces [240], and to avoid obstacles. We follow the setup in P and train policy $\pi_{\text{task}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-terrain}})$, $\boldsymbol{s}_t^{\text{g-terrain}} \triangleq (\boldsymbol{o}_t, \boldsymbol{q}_t{}^{\tau})$ where $\boldsymbol{o}_t$ represents the height map of the humanoid's surrounding, and $\boldsymbol{p}_{t+1}^{\tau}$ is the next 10 time-step's 2D trajectory to follow. The reward is computed as $r_t^{\text{terrain}} = \exp(-2\|\boldsymbol{p}_t{}^{(0)} - \boldsymbol{p}_t^{\tau}\|) - 0.0005 \cdot \sum_{j \in \text{ joints}} |\boldsymbol{\mu}_j \dot{\boldsymbol{q}}_j|^2$ where the first term is the trajectory following the reward and the second term an energy penalty. Random trajectories are generated procedurally, with a velocity between $[0, 3]$m/s and acceleration between $[0, 2]$ m/s^2. The height map $\boldsymbol{o}_t$ is a rasterized local height map of size $\boldsymbol{o}_t \in \mathcal{R}^{32\times32\times3}$, which captures a 2m $\times$ 2m square centered at the humanoid. We do not consider any shape variation or human-to-human interaction as in PACER. Different from PACER, which relies on an additional adversarial reward to achieve realistic and human-like behavior, our framework policy does not rely on any additional reward but can still solve this challenging task with human-like movements. We hypothesize that this is a result of sampling from the pre-learned prior, where human-like motor skills are easier to sample than unnatural ones.

VR Controller Tracking. Tracking VR controllers is the task of inferring full-body human motion from the three 6DOF poses provided by the VR controllers (headset and two hand controllers). Following QuestSim [295], we train this tracking policy using synthetic data. Essentially, we treat the humanoid's head and hand positions as a proxy for headset and controller positions. One can view the VR controller tracking task as an imitation task, but with only three joints to track, with the goal state being: $\boldsymbol{s}_t^{\text{g-vr}} \triangleq (\hat{\boldsymbol{\theta}}_{t+1}^{\text{vr}} \ominus \boldsymbol{\theta}_t{}^{\text{vr}}, \hat{\boldsymbol{p}}_{t+1}^{\text{vr}} - \boldsymbol{p}_t{}^{\text{vr}}, \hat{\boldsymbol{v}}_{t+1}^{\text{vr}} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_t^{\text{vr}} - \boldsymbol{\omega}_t^{\text{vr}}, \hat{\boldsymbol{\theta}}_{t+1}^{\text{vr}}, \hat{\boldsymbol{p}}_{t+1}^{\text{vr}})$ where the superscript $^{\text{vr}}$ refers to selecting only the head and two hands joints. During training, we use the same (full-body) imitation reward to train the policy. We use the same progressive training procedure for training the tracking policy.

5.7 Discussion and Future Work

Failure Cases. While PULSE demonstrated that it is feasible to learn a universal latent space model, it is far from perfect. First, π_{PULSE} does not reach a 100% imitation success rate, meaning that the information bottleneck is a lossy compression of the motor skill generated by PHC+. In our experiments, online distillation can reach 100% success rate using a unconstrained latent space (without the variational information bottleneck). The variational formulation enables probabilistic modeling of motor skills, but introduces additional challenges. For downstream tasks, while our method can lead to human-like behavior, for tasks like VR controller tracking, it can lag behind the performance of training from scratch. For motion generation, the humanoid can be stuck in a fallen or standing position, although increasing the noise into the system could jolt it out of these states.

Future Work. In conclusion, we present PULSE, a universal motion representation for physics-based humanoid control. It can serve as a generative motion prior for downstream generative and estimation tasks alike, without the use of any additional modification or adaptive layers. In the next chapter, we utilize PULSE for dexterous whole-body locomanipulation tasks.

Part III

Dexterous and Perceptive Policy Learning

Chapter 6

Omnigrasp: Grasping Diverse Objects with Simulated Humanoids

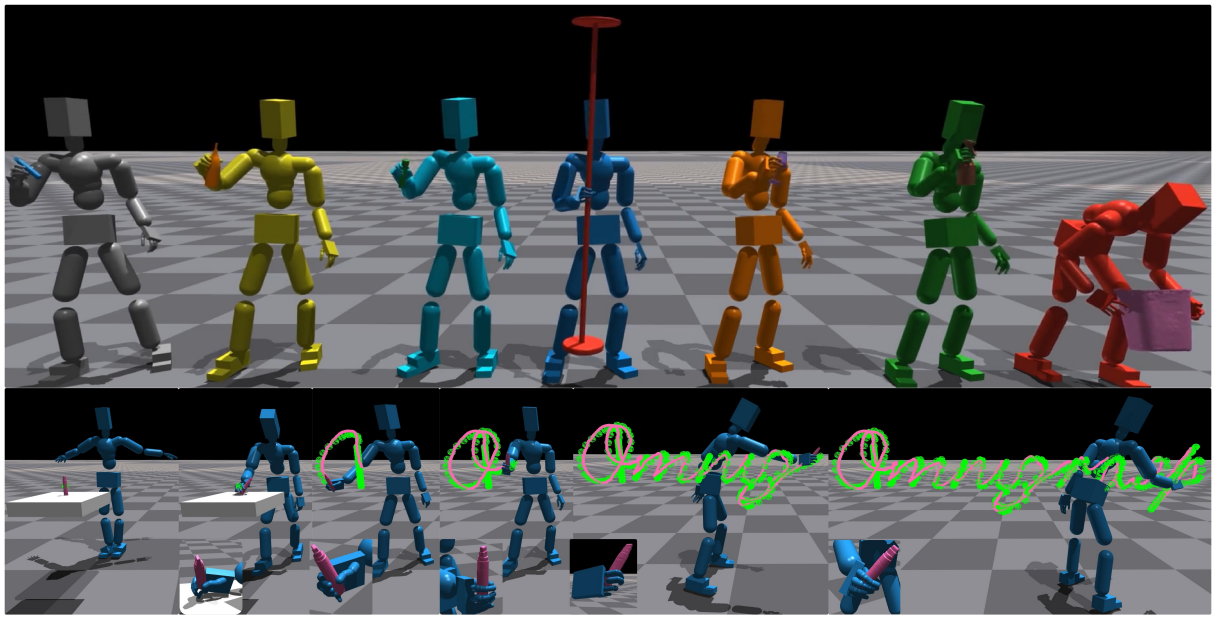


Figure 6.1. We control a simulated humanoid to grasp diverse objects and follow complex trajectories. (*Top*): picking up and holding objects. (*Bottom*): green dots - reference trajectory; pink dots - object trajectory.

6.1 Introduction

Given an object mesh, we aim to control a simulated humanoid equipped with two dexterous hands to pick up the object and follow plausible trajectories, as shown in Fig.6.1. This capability could be broadly applied to creating human-object interactions for animation and AV/VR, with potential extensions to humanoid robotics [97]. However, controlling a simulated humanoid with dexterous hands for precise object manipulation poses significant challenges. The bipedal humanoid must maintain balance to enable detailed movements of the arms and fingers. Moreover, interacting with objects requires forming stable

grasps that accommodate diverse object shapes. Combining these demands with the inherent difficulties of controlling a humanoid with a high degree of freedom (*e.g.* 153 DoF) significantly complicates the learning process.

These challenges have led previous methods of simulated grasping to employ a disembodied hand [51, 61, 225, 312] to grasp and transport. While this approach can generate physically plausible grasps, employing a floating hand compromises physical realism: the hands' root position and orientation are controlled by invisible forces, allowing it to remain nearly perfectly stable during grasping. Moreover, studying the hand in isolation does not accurately reflect its typical use, which is when it is attached to a mobile and flexible body. A naive approach to supporting hands is to use existing full-body motion imitators [169] to provide body control and train additional hand controllers for grasping. However, the presence of a body introduces instability, limits hand movement, and requires synchronizing the entire body to facilitate finger motion. State-of-the-art (SOTA) full-body imitators also have an average 30mm tracking error for the hands, which can cause the humanoid to miss objects. Due to the above challenges, previous work that studies full-body object manipulations often limits its scope to only one sequence of object interaction [293] and encounters difficulties in trajectory following [26], even when trained with highly specialized motion priors.

Another challenge of grasping is the diversity of the object shapes and trajectories. Each object may require a unique type of grasping, and scaling to thousands of different objects often requires training procedures such as generalist-specialist training [312] or curriculum [287,340]. There is also infinite variability in potential object trajectories, and each trajectory may necessitate precise full-body coordination. Thus, prior work typically focuses on simple trajectories, such as vertical lifting [51,312], or on learning a single, fixed, and pre-recorded trajectory per policy [61]. The flexibility with which humans manipulate objects to follow various trajectories while holding them remains unobtainable for current humanoids, even in simulations.

In this work, we introduce a full-body and dexterous humanoid controller capable of picking up and following diverse object trajectories using Reinforcement Learning (RL). Our proposed method, Omnigrasp, presents a scalable approach that generalizes to unseen object shapes and trajectories. Here, "Omni" refers to following any trajectory in all directions within a reasonable range and grasping diverse objects. Our key insight lies in using a pretrained universal dexterous motion representation as the action space. Directly training a policy on the joint actuation space using RL results in unnatural motion and leads to a severe exploration problem. Exploration noise in the torso can lead to a large deviation in the location of the arm and wrist as the noise propagates through the kinematic chain. This can lead to the humanoid quickly knocking the object away, which hinders training progress. Prior work has explored using a separate body and hand latent space trained using adversarial learning [26]. However, as the adversarial latent space can only cover small-scale and curated datasets, these methods do not achieve a high grasping success rate. The separation of hands and body motion prior also adds complexity to the system. We propose using a unified *universal and dexterous* humanoid motion latent space [170]. Learned from a large-scale human motion database [176], our motion representation provides a compact and efficient action space for RL exploration. We enhance the dexterity of this latent space by incorporating articulated hand motions into the existing body-only human motion dataset.

Equipped with a universal motion representation, our humanoid controller does not require any specialized interaction graph [293,341] to learn human-object interactions. Our input to the policy consists only of object and trajectory-following information and is devoid of any grasp or reference body motion. For training, we use randomly generated trajectories and do not require paired full-body human-object motion data. We also identify the importance of pre-grasps [61] (the hand pose right before grasping) and utilize it in our reward design. The resulting policy can be directly applied to transport new objects without additional processing and achieve a SOTA success rate on following object trajectories captured by Motion Capture (MoCap).

To summarize, our contributions are: (1) we design a dexterous and universal humanoid motion representation that significantly increases sample efficiency and enables learning to grasp with simple yet effective state and reward designs; (2) we show that leveraging this motion representation, one can learn grasping policies with synthetic grasp poses and trajectories, without using any paired full-body and object motion data. (3) we demonstrate the feasibility of training a humanoid controller that can achieve a high success rate in grasping objects, following complex trajectories, scaling up to diverse training objects, and generalizing to unseen objects.

6.2 Related Works

Simulated Humanoid Control. Simulated humanoids can be used to create animations [94,155,208,212, 213,215,297,328,341], estimate full-body pose from sensors [84,108,138,168,172,295,324,325,329], and transfer to real humanoid robots [76,97,98,223,224]. Since there are no ground truth data for joint actuation and physics simulators are often non-differentiable, model-based control [106], trajectory optimization [155,303], and deep RL [46,208] are used instead of supervised learning. Due to its flexibility and scalability, deep RL has been popular among efforts in simulated humanoids, where a policy/controller is trained via trial and error. Most of the previous work on humanoids does not consider articulated fingers, except for a few [15,26,155,184]. A dexterous humanoid controller is essential for humanoids to perform meaningful tasks in simulation and in the real world.

Dexterous Manipulation. Dexterous manipulation is an essential topic in robotics [27,28,40,41,50,51, 73,156,225,287,312,330,334,335] and animation [8,26,142,340]. This task usually involves pick-and-place [27,28], lifting [287,312,334], articulating objects [335], and following predefined object trajectories [26,31,61]. Most of these efforts use a disembodied hand for grasping and employ non-physical virtual forces to control the hand. Among them, D-Grasp [51] leverages the MANO [237] hand model for physically plausible grasp synthesis and 6DoF target reaching. UniDexGrasp [312] and its followup [287] use the Shadow Hand [5]. PGDM [61] trains a grasping policy for individual object trajectories and identifies pre-grasp initialization (initializing the hand in a pose right before grasping) as a crucial factor for successful grasping. For the works that consider both hands and body, PMP [15] and PhysHOI [293] train one policy for each task or object. Braun *et al*. [26] studies a similar setting to ours but relies on MoCap human-object interaction data and only uses one hand. Compared to prior work, Omnigrasp trains one policy to transport diverse objects, supports bimanual motion, and achieves a high success rate in lifting and object trajectory following.

Kinematic Grasp Synthesis. Synthesizing hand grasp can be widely applied in robotics and animation. A line of work [25,35,35,72,80,157,178,190,304,321] focuses on reconstructing and predicting grasp from images or videos, while others [192,322] study hand grasp generation to help image generation. Among them, Manipnet and CAMS [332] predict finger poses given a hand object trajectory. TOCH [344] and GeneOH [159] denoise dynamic hand pose predictions for object interactions. More research in this area focuses on generating static or sequential hand poses with a given object as the condition [120, 266,320]. For synthesizing body and hand poses jointly, there are limited MoCap data available [266] due to difficulties in capturing synchronized full-body and object trajectories. Some generative methods [83,144,265,267,273,301,323] can create paired human-object interactions, but they require initialization from the ground truth [83,265,301], or only predict static full-body grasps [273]. In this work, we use GrabNet [266] trained on object shapes from OakInk [316] to generate hand poses as reward guidance for our policy training.

Humanoid Motion Representation. Due to the high DoF of a humanoid and the sample inefficiency of RL training, the search space within which the policy operates during trial and error is crucial. A more structured action space such as motion primitives [89,93,183,229] or motion latent space [213,275] can

significantly increase sample efficiency since the policy can sample coherent motion instead of relying on random "jittering" noise. This is especially important for humanoids with dexterous hands, where the torso motion can drastically affect the hand movement and lead to the humanoid knocking the object away. Thus, prior work in this space utilizes part-based motion priors [15, 26] trained on specialized datasets. While effective in the single task setting where the humanoid only needs to perform actions close to the ones in the specialized datasets, these motion priors can hardly scale to more free-formed motion, such as following randomly generated object trajectories. We extend the recently proposed universal humanoid motion representation, PULSE [170], to the dexterous humanoid setting and demonstrate that a 48-dimensional, full-body-and-hand motion latent space can be used to pick up and follow randomly generated trajectories.

6.3 Preliminaries

We define the human pose as $\boldsymbol{q}_t \triangleq (\boldsymbol{\theta}_t, \boldsymbol{p}_t)$, consisting of 3D joint rotation $\boldsymbol{\theta}_t \in \mathbb{R}^{J\times 6}$ and position $\boldsymbol{p}_t \in \mathbb{R}^{J\times 3}$ of all J links on the humanoid (hands and body), using the 6 degree-of-freedom (DOF) rotation representation [346]. To define velocities $\dot{\boldsymbol{q}}_{1:T}$, we have $\dot{\boldsymbol{q}}_t \triangleq (\boldsymbol{\omega}_t, \boldsymbol{v}_t)$ as angular $\boldsymbol{\omega}_t \in \mathbb{R}^{J\times 3}$ and linear velocities $\boldsymbol{v}_t \in \mathbb{R}^{J\times 3}$. For objects, we define their 3D trajectories $\boldsymbol{q}_t^{\text{obj}}$ using object position $\boldsymbol{p}_t^{\text{obj}}$, orientation $\boldsymbol{\theta}_t^{\text{obj}}$, linear velocity $\boldsymbol{v}_t^{\text{obj}}$, and angular velocity $\boldsymbol{\omega}_t^{\text{obj}}$. As a notation convention, we use $\hat{\cdot}$ to denote the kinematic quantities from Motion Capture (MoCap) or trajectory generator and normal symbols without accents for values from the physics simulation. $\hat{\boldsymbol{O}}$ refers to a dataset of diverse object meshes.

Goal-conditioned Reinforcement Learning for Humanoid Control. We define the object grasping and transporting task using the general framework of goal-conditioned RL. Namely, a goal-conditioned policy π is trained to control a simulated humanoid to grasp an object and follow object trajectories $\hat{\boldsymbol{q}}_{1:T}^{\text{obj}}$ using dexterous hands. The learning task is formulated as a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$ of states, actions, transition dynamics, reward function, and discount factor. The simulation determines the state $\boldsymbol{s}_t \in \mathcal{S}$ and transition dynamics $\mathcal{T}$, where a policy computes the action $\boldsymbol{a}_t$. The state $\boldsymbol{s}_t$ contains the proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$. Proprioception is defined as $\boldsymbol{s}_t^{\text{p}} \triangleq (\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{c}_t)$, which contains the 3D body pose $\boldsymbol{q}_t$, velocity $\dot{\boldsymbol{q}}_t$, and contact forces $\boldsymbol{c}_t$ on the hand. The goal state $\boldsymbol{s}_t^{\text{g}}$ is defined based on the states of the objects. When computing the states $\boldsymbol{s}_t^{\text{g}}$ and $\boldsymbol{s}_t^{\text{p}}$, all values are normalized with respect to the humanoid heading (yaw). Based on proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$, we define a reward $r_t = \boldsymbol{\mathcal{R}}(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ for training the policy. We use proximal policy optimization (PPO) [245] to maximize discounted reward $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$. Our humanoid follows the kinematic structure of SMPL-X [204] using the mean shape. It has 52 joints, of which 51 are actuated. 21 joints are body joints, and the remaining 30 joints are for two hands. All joints have 3 DoF, resulting in an actuation space of $\boldsymbol{a}_t \in \mathbb{R}^{51\times 3}$. Each degree of freedom is actuated by a proportional derivative (PD) controller, and the action $\boldsymbol{a}_t$ specifies the PD target.

6.4 Omnigrasp: Grasping Diverse Objects and Follow Object Trajectories

To tackle the challenging problem of picking up objects and following diverse trajectories, we first acquire a universal dexterous humanoid motion representation in Sec.6.4.1. Using this motion representation,

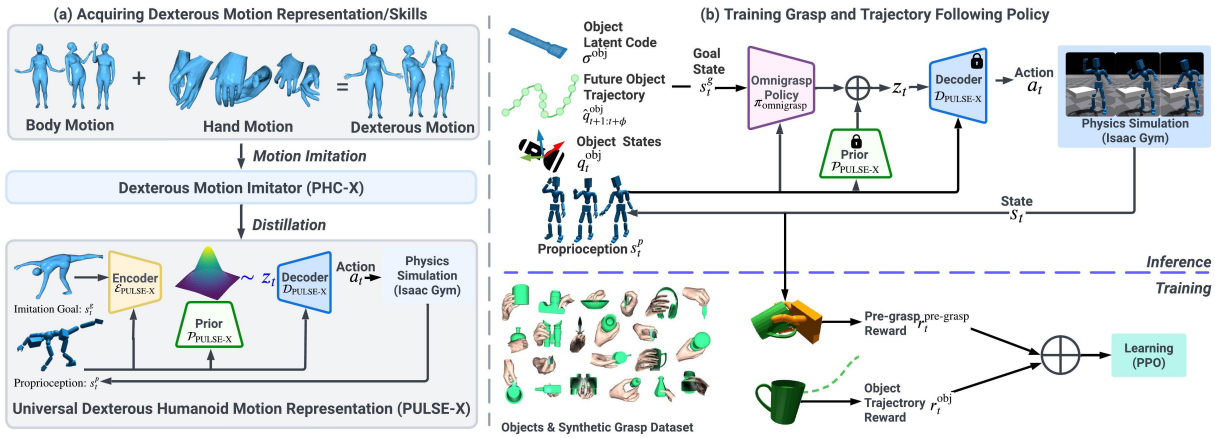


Figure 6.2. Omnigrasp is trained in two stages. (a) A universal and dexterous humanoid motion representation is trained via distillation. (b) Pre-grasp guided grasping training using a pretrained motion representation.

we design a hierarchical RL framework (Sec. 6.4.2) for grasping objects using simple[1] state and reward designs guided by pre-grasps. Our architecture is visualized in Figure 6.2.

6.4.1 PULSE-X: Physics-based Universal Dexterous Humanoid Motion Representation

We introduce PULSE-X that extends PULSE [170] to the dexterous humanoid by adding articulated fingers. We first train a humanoid motion imitator [169] that can scale to a large-scale human motion dataset with finger motion. Then, we distill the motion imitator into a motion representation using a variational information bottleneck (similar to a VAE [129]).

Data Augmentation. Since full-body motion datasets that contain finger motion are rare (*e.g.*, 91% of the AMASS sequences do not have finger motion), we first augment existing sequences with articulated finger motion and construct a dexterous full-body motion dataset. Similarly to the process in BEDLAM [22], we randomly pair full-body motion from AMASS [176] with hand motion sampled from GRAB [266] and Re:InterHand [189] to create a dexterous AMASS dataset. Intuitively, training on this dataset increases the dexterity of the imitator and the subsequent motion representation.

PHC-X: Humanoid Motion Imitation with Articulated Fingers. Inspired by PHC [169], we design PHC-X $\pi_{\text{PHC-X}}$ for humanoid motion imitation with articulated fingers. For the finger joints, *we treat them similarly as the rest of the body* (e.g. *toe or wrist*) and find this formulation sufficient to acquire the dexterity needed for grasping. Formally, the goal state for training $\pi_{\text{PHC-X}}$ with RL is $\boldsymbol{s}_t^{\text{g-mimic}} \triangleq (\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_{t+1} - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_{t+1}, \hat{\boldsymbol{p}}_{t+1})$, which contains the difference between proprioception and one frame reference pose $\hat{\boldsymbol{q}}_{t+1}$.

Learning Motion Representation via Online Distillation. In PULSE [170], an encoder $\boldsymbol{\mathcal{E}}_{\text{PULSE}}$, decoder $\boldsymbol{\mathcal{D}}_{\text{PULSE}}$, and prior $\boldsymbol{\mathcal{P}}_{\text{PULSE}}$ are learned to compress motor skills into a latent representation. For downstream tasks, the frozen decoder and prior will translate the latent code to joint actuation. Formally, the encoder $\boldsymbol{\mathcal{E}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$ computes the latent code distribution based on current input states. The decoder $\boldsymbol{\mathcal{D}}_{\text{PULSE}}(\boldsymbol{a}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t)$ produces action (joint actuation) based on the latent code $\boldsymbol{z}_t$. The prior $\boldsymbol{\mathcal{P}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}})$ defines a Gaussian distribution based on proprioception and replaces the unit Gaussian distribution used in VAEs [129]. The prior increases the expressiveness of the latent space and guides

[1]Here, the "simple reward" refers to not needing paired full-body-and-hand MoCap data when computing the reward, which increases complexity.

downstream task learning by forming a residual action space (see Sec.6.4.2). We model the encoder and prior distribution as diagonal Gaussian:

$$\boldsymbol{\mathcal{E}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}) = \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e), \boldsymbol{\mathcal{P}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}) = \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p). \tag{6.1}$$

To train the models, we use online distillation similar to DAgger [239] by rolling out the encoder-decoder in simulation and querying $\pi_{\text{PHC-X}}$ for action labels $\boldsymbol{a}_t^{\text{PHC-X}}$. For more information and evaluation of PHC-X and PULSE-X, please refer to the Section 6.6.2.

6.4.2 Pre-grasp Guided Object Manipulation

Using hierarchical RL and PULSE-X's trained decoder $\boldsymbol{\mathcal{D}}_{\text{PULSE}}$ and prior $\boldsymbol{\mathcal{P}}_{\text{PULSE}}$, the action space for our object manipulation policy becomes the latent motion representation $\boldsymbol{z}_t$. Since the action space serves as a strong human-like motion prior, we can use simple state and reward design and do not require any paired object and human motion to learn grasping policies. We use only hand pose before grasping (pregraps), either from a generative method or MoCap, to train our policy.

State. To provide the task policy $\pi_{\text{Omnigrasp}}$ with information about the object and the desired object trajectory, we define the goal state as

$$\boldsymbol{s}_t^{\text{g}} \triangleq (\hat{\boldsymbol{p}}_{t+1:t+\phi}^{\text{obj}} - \boldsymbol{p}_t^{\text{obj}}, \hat{\boldsymbol{\theta}}_{t+1:t+\phi}^{\text{obj}} \ominus \boldsymbol{\theta}_t^{\text{obj}}, \hat{\boldsymbol{v}}_{t+1:t+\phi}^{\text{obj}} - \boldsymbol{v}_t^{\text{obj}}, \hat{\boldsymbol{\omega}}_{t+1:t+\phi}^{\text{obj}} - \boldsymbol{\omega}_t^{\text{obj}}, \boldsymbol{p}_t^{\text{obj}}, \boldsymbol{\theta}_t^{\text{obj}}, \boldsymbol{\sigma}^{\text{obj}}, \boldsymbol{p}_t^{\text{obj}} - \boldsymbol{p}_t^{\text{hand}}), \tag{6.2}$$

which contains the reference object pose and the difference between the reference object trajectory for the next ϕ frames and the current object state. $\boldsymbol{\sigma}^{\text{obj}} \in \mathcal{R}^{512}$ is the object shape latent code computed using the canonical object pose and Basis Point Set (BPS) [219]. $\boldsymbol{p}_t^{\text{obj}} - \boldsymbol{p}_t^{\text{hand}}$ is the difference between the current object position and each hand joint position. All values are normalized with respect to the humanoid heading. Notice that the state $\boldsymbol{s}_t^{\text{g}}$ does not contain body pose, grasp, or phase variables [26], which makes our method applicable to unseen objects and reference trajectories at test time.

Action. Similar to downstream task policies in PULSE, we form the action space of $\pi_{\text{Omnigrasp}}$ as the residual action with respect to prior's mean $\boldsymbol{\mu}_t^p$ and compute the PD target $\boldsymbol{a}_t$:

$$\boldsymbol{a}_t = \boldsymbol{\mathcal{D}}_{\text{PULSE}}(\pi_{\text{Omnigrasp}}(\boldsymbol{z}_t^{\text{omnigrasp}}|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}}) + \boldsymbol{\mu}_t^p), \tag{6.3}$$

where $\boldsymbol{\mu}_t^p$ is computed by the prior $\boldsymbol{\mathcal{P}}_{\text{PULSE}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}})$. The policy $\pi_{\text{Omnigrasp}}$ computes $\boldsymbol{z}_t^{\text{omnigrasp}} \in \mathcal{R}^{48}$ instead of the target $\boldsymbol{a}_t \in \mathcal{R}^{51\times 3}$ directly, and leverages the latent motion representation of PULSE-X to produce human-like motion.

Reward. While our policy does not take any grasp guidance or reference body trajectory *as input*, we utilize pre-grasp guidance in the *reward*. We refer to pre-grasp $\hat{\boldsymbol{q}}^{\text{pre-grasp}} \triangleq (\hat{\boldsymbol{p}}^{\text{pre-grasp}}, \hat{\boldsymbol{\theta}}^{\text{pre-grasp}})$ as a single frame of hand pose consisting of hand translation $\hat{\boldsymbol{p}}^{\text{pre-grasp}}$ and rotation $\hat{\boldsymbol{\theta}}^{\text{pre-grasp}}$. PGDM [61] shows that initializing a floating hand to pre-grasps can help the policy better reach objects and initiate manipulation. As we do not initialize the humanoid with the pre-grasp pose as in PGDM, we design a stepwise pre-grasp reward:

$$\boldsymbol{r}_t^{\text{omnigrasp}} = \begin{cases} r_t^{\text{approach}}, & \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 > 0.2 \text{ and } t < \lambda \\ r_t^{\text{pre-grasp}}, & \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 \leq 0.2 \text{ and } t < \lambda \\ r_t^{\text{obj}}, & t \geq \lambda, \end{cases} \tag{6.4}$$

based on time and the distance between the object and hands. Here, $\lambda = 1.5s$ indicates the frame in which grasping should occur, and $\boldsymbol{p}_t^{\text{hand}}$ indicates the hand position. When the object is far away from the hands ($\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 > 0.2$), we use an approach reward r_t^{approach} similar to a point-goal [169,299]

Algo 6: Learn Omnigrasp

Function `TrainOmnigrasp`($\mathcal{D}_{PULSE}, \mathcal{P}_{PULSE}, \pi_{PULSE}, \hat{\boldsymbol{O}}, \mathcal{T}^{3D}$)**:**

  Input: Pretrained PULSE-X's decoder $\mathcal{D}_{\text{PULSE}}$ and prior $\mathcal{P}_{\text{PULSE}}$, Object mesh dataset $\hat{\boldsymbol{O}}$, 3D trajectory Generator $\mathcal{T}^{3D}$;

  while *not converged* **do**

    $\boldsymbol{M} \leftarrow \emptyset$ initialize sampling memory ;

    while $\boldsymbol{M}$ *not full* **do**

      $\boldsymbol{q}_0^{\text{obj}}, \hat{\boldsymbol{p}}^{\text{pre-grasp}}, \boldsymbol{s}_t^{\text{p}} \sim$ randomly sample initial object state, pre-grasp, and humanoid state ;

      $\hat{\boldsymbol{q}}_{1:T}^{\text{obj}} \sim$ sample reference object trajectory using $\mathcal{T}^{3D}$;

      for $t \leftarrow 1...T$ **do**

        $\boldsymbol{z}_t^{\text{omnigrasp}} \sim \pi_{\text{PULSE}}(\boldsymbol{z}_t^{\text{omnigrasp}} | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ `// use pretrained latent space as action space;`

        $\boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p \leftarrow \mathcal{P}_{\text{PULSE}}(\boldsymbol{z}_t | \boldsymbol{s}_t^{\text{p}})$ `// compute prior latent code;`

        $\boldsymbol{a}_t \leftarrow \mathcal{D}_{\text{PULSE}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t^{\text{omnigrasp}} + \boldsymbol{\mu}_t^p)$ `// decode action using pretrained decoder;`

        $\boldsymbol{s}_{t+1} \leftarrow \mathcal{T}(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t)$ `// simulation;`

        $\boldsymbol{r}_t \leftarrow \mathcal{R}(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}})$ `// compute reward;`

        store $(\boldsymbol{s}_t, \boldsymbol{z}_t^{\text{omnigrasp}}, \boldsymbol{r}_t, \boldsymbol{s}_{t+1})$ into memory $\boldsymbol{M}$;

    $\pi_{\text{PULSE}} \leftarrow$ PPO update using experiences collected in $\boldsymbol{M}$;

    $\hat{\boldsymbol{O}}_{\text{hard}} \leftarrow$ Eval and pick hard object subset to train on.

  return $\pi_{\text{Omnigrasp}}$;

reward $r_t^{\text{approach}} = \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 - \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_{t-1}^{\text{hand}}\|_2$,, where the policy is encouraged to get close to the pre-grasp. After the hands are close enough (≤ 0.2m), we use a more precise hand imitation reward: $r_t^{\text{pre-grasp}} = w_{\text{hp}} e^{-100\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 \times \mathbb{1}\{\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \hat{\boldsymbol{p}}_t^{\text{obj}}\|_2 \leq 0.2\}} + w_{\text{hr}} e^{-100\|\hat{\boldsymbol{\theta}}^{\text{pre-grasp}} - \boldsymbol{\theta}_t^{\text{hand}}\|_2}$, to encourage the hands to be close to pre-grasps. For grasps that involve only one hand, we use an indicator variable $\mathbb{1}\{\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \hat{\boldsymbol{p}}_t^{\text{obj}}\|_2 \leq 0.2\}$ to filter out hands that are too far away from the object. After timestep λ, we use only the object trajectory following reward:

$$r_t^{\text{obj}} = (w_{\text{op}} e^{-100\|\hat{\boldsymbol{p}}_t^{\text{obj}} - \boldsymbol{p}_t^{\text{obj}}\|_2} + w_{\text{or}} e^{-100\|\hat{\boldsymbol{\theta}}_t^{\text{obj}} - \boldsymbol{\theta}_t^{\text{obj}}\|_2} + w_{\text{ov}} e^{-5\|\hat{\boldsymbol{v}}_t^{\text{obj}} - \boldsymbol{v}_t^{\text{obj}}\|_2} + w_{\text{oav}} e^{-5\|\hat{\boldsymbol{\omega}}_t^{\text{obj}} - \boldsymbol{\omega}_t^{\text{obj}}\|_2}) \cdot \mathbb{1}\{\text{C}\} + \mathbb{1}\{\text{C}\} \cdot w_{\text{c}}. \tag{6.5}$$

r_t^{obj} computes the difference between the current and reference object pose, which is filtered by an indicator variable $\mathbb{1}\{\text{C}\}$ that is set to true if the object is in contact with the humanoid hands. The reward $\mathbb{1}\{\text{C}\} \cdot w_{\text{c}}$ encourages the humanoid's hand to have contact with the object. Hyperparameters can be found in Section 6.6.5.

Object 3D Trajectory Generator. As there is a limited number of ground-truth object trajectories [61], either collected from MoCap or animators, we design a 3D object trajectory generator that can create trajectories with varying speed and direction. Using the trajectory generator, our policy can be trained without any ground-truth object trajectories. This strategy provides better coverage of potential object trajectories, and the resulting policy achieves higher success in following unseen trajectories (see Table 6.1). Specifically, we extend the 2D trajectory generator used in PACER [233, 290] to 3D, and create our trajectory generator $\mathcal{T}^{3D}(\boldsymbol{q}_0^{\text{obj}}) = \hat{\boldsymbol{q}}_{1:T}^{\text{obj}}$. Given initial object pose $\boldsymbol{q}_0^{\text{obj}}$, $\mathcal{T}^{3D}$ can generate a sequence of plausible reference object motion $\hat{\boldsymbol{q}}_{1:T}^{\text{obj}}$. We limit the z-direction trajectory to between 0.03m and 1.8m and leave the xy direction unbounded. For more information and sampled trajectories, please refer to Section 6.6.5.

Training. Our training process is depicted in Algo 6. One of the main sources of performance improvement for motion imitation is hard-negative mining [169, 172], where the policy is evaluated regularly to

		GRAB-Goal-Test (Cross-Object, 140 sequences, 5 unseen objects)							GRAB-IMoS-Test (Cross-Subject, 92 sequences, 44 objects)						
Method	Traj	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
PPO-10B	Gen	98.4%	55.9%	97.5%	36.4	**0.4**	21.0	14.5	96.8%	53.2%	97.9%	35.6	**0.5**	19.6	13.9
PHC [169]	MoCap	3.6%	11.4%	81.1%	66.3	0.8	1.5	3.8	0%	3.3%	97.4%	56.5	0.3	1.4	2.9
AMP [215]	Gen	90.4%	46.6%	94.0 %	40.7	0.6	5.3	5.3	95.8 %	49.2%	96.5%	34.9	0.5	6.2	6.0
Braun *et al.* [26]	MoCap	79%	-	85%	-	-	-	-	64%	-	65%	-	-	-	-
Omnigrasp	MoCap	94.6%	84.8%	98.7%	**28.0**	**0.5**	**4.2**	**4.3**	95.8%	85.4%	99.8%	**27.5**	**0.6**	**5.0**	**5.0**
Omnigrasp	Gen	**100%**	**94.1%**	**99.6%**	30.2	0.93	5.4	4.7	**98.9%**	**90.5%**	**99.8%**	27.9	0.97	6.3	5.4

Table 6.1. Quantitative results on object grasp and trajectory following on the GRAB dataset.

find the failure sequences to train on. Thus, instead of using object curriculum [287, 312, 340], we use a simple hard-negative mining process to pick hard objects $\hat{\boldsymbol{O}}_{\text{hard}}$ to train on. Specifically, let s_j be the number of failed lifts for object j over all previous runs. The probability of choosing object j among all objects is $P(j) = \frac{s_j}{\sum_i^J s_i}$.

Object and Humanoid Initial State Randomization. Since objects can have diverse initial positions and orientations with respect to the humanoid, it is crucial to have the policy exposed to diverse initial object states. Given the object dataset $\hat{\boldsymbol{O}}$ and the provided initial states (either from MoCap or by dropping the object in simulation) $\boldsymbol{q}_0^{\text{obj}}$, we perturb $\boldsymbol{q}_0^{\text{obj}}$ by adding randomly sampled yaw-direction rotation and adjusting the position component $\boldsymbol{q}_0^{\text{obj}}$. We do not change the pitch and yaw of the object's initial pose as some poses are invalid in simulation. For the humanoid, we use the initial state from the dataset if provided (*e.g.* GRAB dataset [266]), and a standing T-pose if there is no paired data.

Inference. During inference, the object latent code $\boldsymbol{p}_t^{\text{obj}}$, a random object starting pose $\boldsymbol{q}_0^{\text{obj}}$, and desired object trajectory $\hat{\boldsymbol{q}}_{1:T}^{\text{obj}}$ is all that is required, without any dependency on pre-grasps or paired kinematic human pose.

6.5 Experiments

Datasets. We use the GRAB [266], OakInk [316], and OMOMO [142] to study grasping small and large objects. The GRAB dataset contains 1.3k paired full-body motion and object trajectories of 50 objects (we remove the doorknob as it is not movable). Since the GRAB dataset provides reference body and object motion, we use them to extract initial humanoid positions and pre-grasps. We follow prior art [26] in constructing cross-object (45 for training and 5 for testing) and cross-subject (9 subjects for training and 1 for testing) train-test sets. On GRAB, we evaluate on following MoCap object trajectories using the mean body shape humanoid. The OakInk dataset contains 1700 diverse objects of 32 categories with real-world scanned and generated object meshes. We split them into 1330 objects for training, 185 for validation, and 185 for testing. Train-test splits are conducted within categories, with train and test splits containing objects from all categories. Since no paired MoCap human motion or grasps exists for the OakInk dataset, we use an off-the-shelf grasp generator [316] to create pre-grasps. The OMOMO dataset contains 15 large objects (table lamps, monitors, *etc.*) with reconstructed mesh, and we pick 7 of them that have cleaner meshes. Due to the limited number of objects from OMOMO, we only test lifting on the objects used for training to verify that our pipeline can learn to move larger objects. On OMOMO and OakInk, we study vertical lifting (30cm) and holding (3s) as the trajectory for quantitative results.

Implementation Details. Simulation is conducted in Isaac Gym [177], where the policy is run at 30 Hz and the simulation at 60 Hz. For PULSE-X and PHC-X, each policy is a 6-layer MLP. For the grasping task, we employ a GRU [49] based recurrent policy and use a GRU with a latent dimension of 512, followed by a 3-layer MLP. We train Omnigrasp for three days collecting around 10^9 samples on a Nvidia A100 GPU. PHC-X and PULSE-X are trained once and frozen, which takes around 1.5 weeks and 3 days. Object

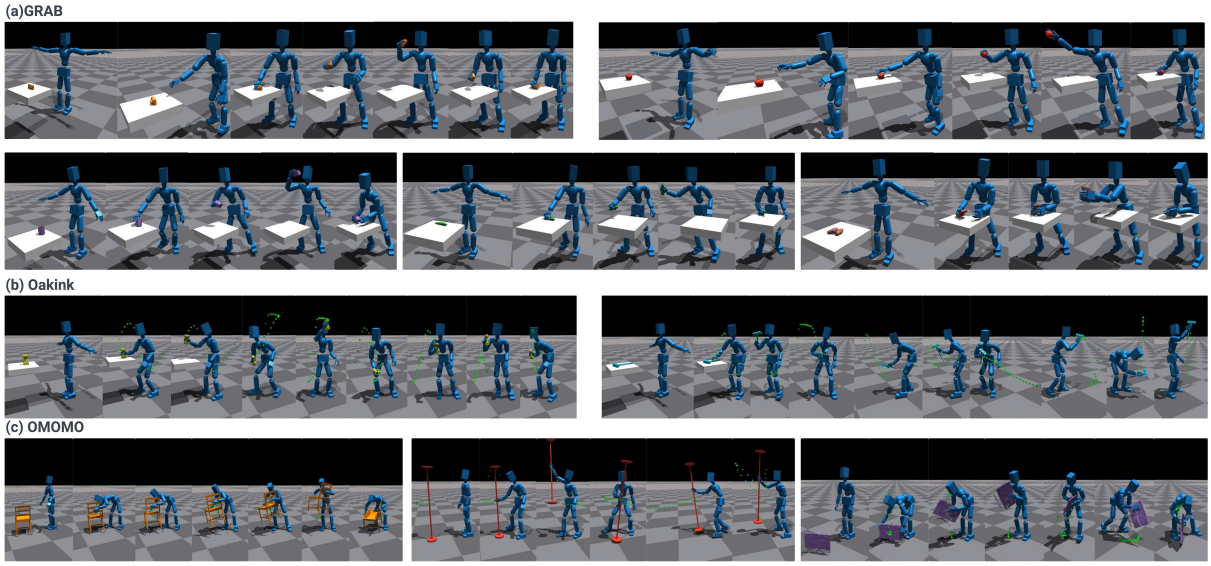


Figure 6.3. Qualitative results. Unseen objects are tested for GRAB and OakInk. Green dots: reference trajectories. Best seen in videos on our supplement site.

density is 1000 kg/m^3. The static and dynamic friction coefficients of the object and humanoid fingers are set to 1. For reference object trajectory, we use $\phi = 20$ future frames sampled at 15Hz. For more details, please refer to Section 6.6.5.

Metrics. For the object trajectory following, we report the position error E_{pos} (mm), rotation error E_{rot} (radian), and physics-based metrics such as acceleration error E_{acc} (mm/frame^2) and velocity error E_{vel} (mm/frame). Following prior art in full-body simulated humanoid grasping [26], we report the grasp success rate $\text{Succ}_{\text{grasp}}$ and Trajectory Targets Reached (TTR). The grasp success rate $\text{Succ}_{\text{grasp}}$ deems a grasp successful when the object is held for at least 0.5s in the physics simulation without dropping. TTR measures the ratio of the target position ($<$ 12cm away from the target position) reached over all the time steps in the trajectory and is only measured on successful trajectories. To measure the complete trajectory success rate, we also report $\text{Succ}_{\text{traj}}$, where a trajectory following is unsuccessful if, at any point in time, the object is $>$ 25cm away from the reference.

6.5.1 Grasping and Trajectory Following

As motion is best seen in videos, please refer to supplement site for extended evaluation on trajectory following, unseen objects, and robustness. Unless otherwise specified, all policies are trained on their respective dataset training split, and we conduct cross-dataset experiments on GRAB and OakInk. All experiments are run 10 times and averaged as the simulator yields slightly different results for each run due to *e.g.* floating-point error. As full-body simulated humanoid grasping is a relatively new task with a limited number of baselines, we use Braun et [26] as our main comparison. We also implement AMP [215] and PHC [169] as baselines. We train AMP with a similar state and reward design (without using PULSE-X's latent space) and a task and discriminator reward weighting of 0.5 and 0.5. PHC refers to using an imitator for grasping, where we directly feed ground-truth kinematic body and finger motion to a pretrained imitator to grasp objects. Since PHC and PULSE-X require pre-training, we also include PPO-10B, which is trained using RL without PULSE-X for a month ($\sim$10 billion samples).

GRAB Dataset (50 objects). Since Braun *et al.* do not use randomly generated trajectories, we train Omnigrasp using two different settings for a fair comparison: one trained with MoCap object trajectories only, and one trained using synthetic trajectories only. From Table 6.1, we can see that our method

	OakInk-Train (1330 objects)							OakInk-Test (185 objects)						
Training Data	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
OakInk	93.7%	86.2%	**100%**	21.3	**0.4**	7.7	6.0	**94.3%**	87.5%	**100%**	**21.2**	**0.4**	7.6	5.9
GRAB	84.5%	75.2%	99.9%	22.4	**0.4**	6.8	5.7	81.9%	72.1%	99.9%	22.7	**0.4**	7.1	5.8
GRAB + OakInk	**95.6%**	**92.0%**	**100%**	**21.0**	0.6	**5.4**	**4.8**	93.5%	**89.0%**	**100%**	21.3	0.6	**5.4**	**4.8**

Table 6.2. Quantitative results on OakInk with our method. We also test Omnigrasp cross-dataset, where a policy trained on GRAB is tested on the OakInk dataset.

outperforms prior SOTA and baselines on all metrics, especially on success rate and trajectory following. Since all methods are simulation-based, we omit penetration/foot sliding metrics and report the precise trajectory tracking errors instead. Training directly using PPO without PULSE-X leads to a performance that significantly lags behind Omnigrasp, even though it has used similar aggregate samples (counting PHC-X and PULSE-X training). Compared to Braun *et al*., Omnigrasp achieves a high success rate on both object lifting and trajectory following. Directly using the motion imitator, PHC, yields a low success rate even when the ground-truth kinematic pose is provided, showing that the imitator's error (on average 30mm) is too large to overcome for precise object grasping. The body shape mismatch between MoCap and our simulated humanoid also contributes to this error. AMP leads to a low trajectory success rate, showing the importance of using a motion prior in the *actions space*. Omnigrasp can track the MoCap trajectory precisely with an average error of 28mm. Comparing training on MoCap trajectories and randomly generated ones, we can see that training on generated trajectories achieves better performance on success rate and position error, though worse on rotation error. This is due to our 3D trajectory generator offering good coverage on physically plausible 3D trajectories, but there is a gap between the randomly generated rotations and MoCap object rotation. This can be improved by introducing more rotation variation on the trajectory generator. The gap between trajectory $\text{Succ}_{\text{traj}}$ and grasp success $\text{Succ}_{\text{grasp}}$ shows that following the full trajectory is a much harder task than just grasping, and the object can be dropped during trajectory following. Qualitative results can be found in Fig. 6.3.

OakInk Dataset (1700 objects). On the OakInk dataset, we scale our grasping policy to >1000 objects and test our generalization to unseen objects. We also conduct cross-dataset experiments, where we train on the GRAB dataset and test on the OakInk dataset. Results are shown in Table 6.2. We can see that 1272 out of the 1330 objects are trained to be picked up, and the whole lifting process also has a high success rate. We observe similar results on the test split. Upon inspection, the failed objects are usually either too large or too small for the humanoid to establish a grasp. The large number of objects also places a strain on the hard-negative mining process. The policy trained on both GRAB and OakInk shows the highest success rate, as on GRAB, there are bi-manual pre-grasps, and the policy learned to use both hands. Using both hands significantly improves the success rate on some larger objects, where the humanoid can scoop up the object with one hand and carry it with both. As OakInk only has pre-grasps using one hand, it cannot learn such a strategy. Surprisingly, training on only GRAB achieves a high success rate on OakInk, picking up more than 1000 objects without training on the dataset, showcasing the robustness of our grasping policy on unseen objects.

OMOMO (7 objects)						
$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
7/7	7/7	100%	22.8	0.2	3.1	3.3

Table 6.3. Quantitative results on the OMOMO dataset.

OMOMO Dataset (7 objects). On the OMOMO dataset, we train a policy to show that our method can learn to pick up large objects. Table 6.3 shows that our method can successfully learn to pick up all the objects, including chairs and lamps. For larger objects, the pre-grasp guidance is essential for guiding the policy to learn bi-manual manipulation skills (as is shown in Fig 6.3)

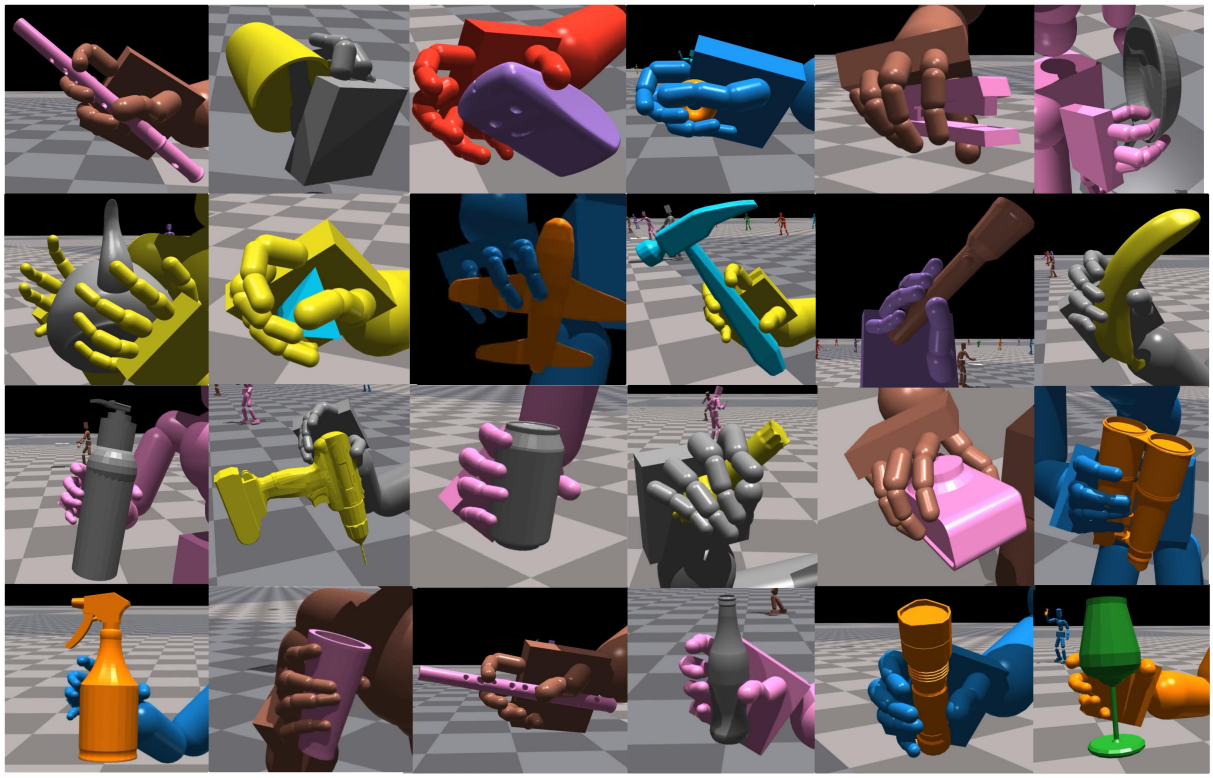


Figure 6.4. Omnigrasp uses diverse strategies to grasp objects of different shapes.

6.5.2 Ablation and Analysis

Ablation. In this section, we study the effects of different components of our framework using the cross-object split of the GRAB dataset. Results are shown in Table 6.4. First, we compare our method trained with (Row 6) or without (R1) PULSE-X's action space. Using the same reward and state design, we can see that using the universal motion prior significantly improves success rates. Upon inspection, using PULSE-X also yields human-like motion, while not using it leads to unnatural motion (see in `supplement site`). R2 vs. R6 shows that the pre-grasp guidance is essential in learning grasps that are stable for grasping objects, but without it, some objects can still be grasped successfully. The difference between R3 and R6 is whether to train using the dexterous AMASS dataset. R3 vs R6 shows that without training on a dataset that has diverse hand motion and full-body motion, the policy can learn to pick up objects (high grasp success rate), but struggles in trajectory following. This is expected as the motion prior probably lacks the motion of "holding the object while moving". R4 and R5 show that object position randomization and hard-negativing mining are crucial for learning robust and successful policies. Ablations on the object latent code, RNN policy, *etc*. can be found in the Section 6.6.5.

Analysis: Diverse Grasps. In Fig. 6.5, we visualize the grasping strategy used by our method. We can see that based on the object shape, our policy uses a diverse set of grasping strategies to hold the object during the trajectory following. Based on the trajectory and object initial pose, Omnigrasp discovers different grasping poses for the *same* object, showcasing the advantage of using simulation and laws of physics for grasp generation. We also notice that for larger objects, our policy will resort to using two hands and a non-prehensile transport strategy. This behavior is learned from pre-grasps in GRAB, which utilize both hands for object manipulation.

Analysis: Robustness and Potential for Sim-to-real Transfer. In Table 6.5, we add uniform random

GRAB-Goal-Test (Cross-Object, 140 sequences, 5 unseen objects)												
idx	PULSE-X	pre-grasp	Dex-AMASS	Rand-pose	Hard-neg	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR ↑	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
1	✗	✓	✓	✓	✓	97.0%	33.6%	92.8%	43.5	**0.5**	10.6	8.3
2	✓	✗	✓	✓	✓	77.1%	57.9%	97.4%	54.9	1.0	5.5	5.2
3	✓	✓	✗	✓	✓	94.4%	77.3%	99.3%	30.5	0.9	4.8	**4.4**
4	✓	✓	✓	✗	✓	92.9%	79.9%	99.2%	31.4	1.1	**4.5**	**4.4**
5	✓	✓	✓	✓	✗	94.0%	71.6%	98.4%	32.3	1.3	6.2	5.7
6	✓	✓	✓	✓	✓	**100**%	**94.1%**	**99.6%**	**30.2**	0.9	5.4	4.7

Table 6.4. Ablation on various strategies of training Omnigrasp. PULSE-X: whether to use the latent motion representation. pre-grasp: pre-grasp guidance reward. Dex-AMASS: whether to train PULSE-X on the dexterous AMASS dataset. Rand-pose: randomizing the object initial pose. Hard-neg: hard-negative mining.

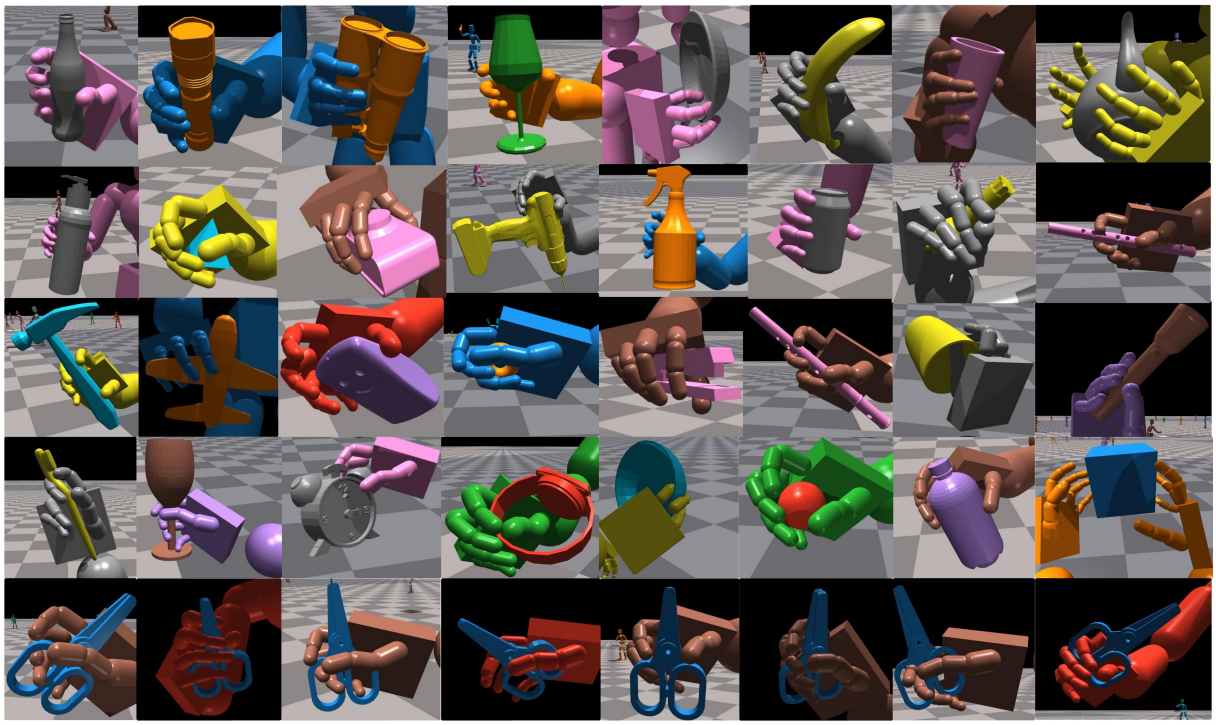


Figure 6.5. (*Top rows*): grasping different objects using both hands. (*Bottom*) diverse grasps on the same object.

noise [-0.01, 0.01] to both task observation (positions, object latent codes, etc.) and proprioception. A similar scale (0.01) of random noise is used in sim-to-real RL to tackle noisy input in real-world humanoids [98]. We see that Omnigrasp is relatively robust to input noise, even though it has not been trained with noisy input. Performance drop is more prominent in the acceleration and velocity metrics. Adding noise during training can further improve robustness. We do not claim that Omnigrasp is currently ready for real-world deployment, but we believe that a similar system design plus sim-to-real modifications (e.g. domain randomization, distilling into a vision-based policy) has the potential. We conduct more analysis on the robustness of our method with respect to initial object position, object weight, and object trajectories on our `supplement site`.

		GRAB-Goal-Test (Cross-Object, 140 sequences, 5 unseen objects)							GRAB-IMoS-Test (Cross-Subject, 92 sequences, 44 objects)						
Method	Noise Scale	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
Omnigrasp	0	**100%**	**94.1%**	**99.6%**	**30.2**	**0.93**	**5.4**	**4.7**	98.9%	**90.5%**	**99.8%**	**27.9**	**0.97**	**6.3**	**5.4**
Omnigrasp	0.01	**100%**	91.4%	99.2%	34.8	1.1	15.6	11.5	**99.5%**	86.2%	99.6%	32.5	1.0	17.9	13.2

Table 6.5. Study on how noise affects pretrained Omnigrasp Policy

6.6 Additional Details

6.6.1 Qualitative Results

Extensive qualitative results are provided at the project page as well as the supplementary zip files (which contain lower-resolution videos due to file size limitations). As motion is best seen in videos, we highly encourage our readers to view them to judge the capabilities of our method better. Specifically, we visualize using our controller to trace the characters "Omnigrasp" in the air while holding unseen objects during training. This complex trajectory is never seen during training. We also visualize the policy on GRAB [266], OakInk [316], and OMOMO [142] datasets, both for training and testing objects. On the GRAB dataset, we follow MoCap trajectories, while for the OakInk and OMOMO datasets, we showcase randomly generated trajectories for training. To demonstrate robustness to different object poses, weights, and directions, we also test our method by varying these variables and show that it can still pick up objects. Interestingly, we notice that our method prefers to use both hands to pick and hold the object as the weight of the object increases. We also include motion imitation and random motion sampling for PHC-X and PULSE-X. Further, we visualize our constructed dexterous AMASS dataset and the motion imitation result. Last, we include failure cases for grasping and trajectory following.

6.6.2 Details about PHC-X and PULSE-X

Data Cleaning. To train both PHC-X and PULSE-X, we follow PULSE's [170] procedure in filtering on implausible motion. This process yields 14889 motion sequences from the AMASS dataset for training our humanoid motion representation. Out of all 14889 sequences, only 9% of the sequences contain hand motion, and training on it will bias the motion imitator to have limited dexterity. Thus, we construct the dexterous AMASS dataset by pairing hand-only motion with body-only motion and demonstrate its effectiveness in learning a motion representation that enables object grasping.

6.6.3 Training and Architecture

The state, action, and rewards for PHC-X and PULSE-X follow the implementation choices of PULSE with the only modifications on the training data (dexterous AMASS) and humanoid (SMPL-X). PHC-X is trained for 1.5 week while PULSE-X takes 3 days. We use the same-sized networks: 6-layer MLP of units [2048, 1536, 1024, 1024, 512, 512] for PHC-X and 3-layer MLP of units [3096, 2048, 1024] for PULSE-X's encoder and decoders. We notice that due to the increase in DoF from SMPL (69) to SMPL-X (153), simulation is $\sim$2 times slower.

6.6.4 Evaluation

We evaluate PULSE-X and PHC-X on our constructed dexterous AMASS dataset. The metrics we use are the mean per-joint position error (mm) for both global $E_{\text{g-mhpe}}$ and local E_{mpjpe} (root-relative) settings. We also report acceleration and velocity errors, similar to the object trajectory following the setting but averaged across all body joints. From Table 6.6, we can see that PHC-X and PULSE-X achieve a high success rate on training data while maintaining a low per-joint error. Distilling from PHC-X to PULSE-X,

	Dexterous AMASS-Train				
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
PHC-X	99.9 %	29.4	31.0	4.1	5.1
PULSE-X	99.5 %	42.9	46.4	4.6	6.7

Table 6.6. Imitation result on dexterous AMASS (14889 sequences).

Method	Batch Size	Learning Rate	σ	γ	ϵ						# of samples
PHC-X	3072	2×10^{-5}	0.05	0.99	0.2						$\sim 10^{10}$
	Batch Size	Learning Rate	Latent size								# of samples
PULSE-X	3072	5×10^{-4}	48								$\sim 10^{9}$
	Batch Size	Learning Rate	σ	γ	ϵ	w_{op}	w_{or}	w_{ov}	w_{oav}	w_{c}	# of samples
Omnigrasp	3072	5×10^{-4}	0.36	0.99	0.2	0.5	0.3	0.05	0.05	0.1	$\sim 10^{9}$

Table 6.7. Hyperparameters for Omnigrasp, PHC-X, and PULSE-X. σ: fixed variance for policy. γ: discount factor. ϵ: clip range for PPO.

we observe similar degradation in imitation performance as in PULSE, akin to the reconstruction error in training VAEs [129].

6.6.5 Details about Omnigrasp

6.6.6 Object Processing

Since the simulator requires convex objects for simulation, we use the built-in v-hacd function to decompose the meshes into convex geometries. The parameters we use for decomposition can be found in Table 6.7. To compute object latent code, we use 512-d BPS [219] by randomly sampling 512 points on a unit sphere and calculating their distances to points on the object mesh. As some object meshes have a large number of vertices, we also perform quadratic decimation on the mesh if it contains more than 50000 vertices.

6.6.7 Training Details

Early Termination. During training, we terminate the episode whenever the object is more than 12cm away from its desired reference trajectory at time step t: $\|\hat{\boldsymbol{p}}_t^{\text{obj}} - \boldsymbol{p}_t^{\text{obj}}\|_2 > 0.12$.

Table Removal. Since the GRAB and OakInk datasets are table-top objects, we use a table at the beginning of the episode to support the object. However, since our randomly generated trajectory can collide with the table and the humanoid has no environmental awareness except for the object, we remove the table after certain timestamps (1.5s) during training.

Contact Detection. As IsaacGym does not provide easy access to contact labels and only provides contact forces, there is no way of differentiating between contact with the table, humanoid body, or objects. Thus, we resort to a heuristic-based way to detect contact. Specifically, if the object is within 0.2m from the hands, has non-zero contact forces, and has a non-zero velocity, we deem it to have contact with the hands.

Trajectory Generator. Randomly generated trajectories can be seen on our `supplement site` on the OakInk and OMOMO dataset, as there is no paired MoCap object motion for these datasets. We sample a random velocity and delta angle at each time step and aggregate the velocities to produce full trajectories.

GRAB-Goal-Test (Cross-Object, 140 sequences, 5 unseen objects)										
idx	Object Latent	RNN	Im-obs	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$E_{\text{pos}} \downarrow$	$E_{\text{rot}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
1	✗	✓	✗	**100%**	93.2%	**99.8%**	**28.7**	1.3	6.1	5.1
2	✓	✗	✗	99.9%	89.6%	99.0%	33.4	1.2	4.5	4.4
3	✓	✓	✓	95.2	77.8%	97.9%	32.2	0.9	**3.2**	**3.9**
4	✓	✓	✗	**100%**	**94.1%**	99.6%	30.2	**0.9**	5.4	4.7

Table 6.8. Additional ablations: Object-latent refers to whether to provide the object shape latent code $\boldsymbol{\sigma}^{\text{obj}}$ to the policy. RNN refers to either using an RNN-based policy or an MLP-based policy. Im-obs refers to whether to provide the policy with ground truth full-body pose $\hat{\boldsymbol{q}}_{t+1}$ as input.

Object	Braun *et al.* [26]			PULSE-X		
	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$	$\text{Succ}_{\text{grasp}} \uparrow$	$\text{Succ}_{\text{traj}} \uparrow$	TTR $\uparrow$
Apple	95%	-	91%	**100%**	99.6%	**99.9%**
Binoculars	54%	-	83%	**100%**	90.5%	**99.6%**
Camera	95%	-	85%	**100%**	97.7%	**99.7%**
Mug	89%	-	74%	**100%**	97.3%	**99.8%**
Toothpaste	64%	-	94%	**100%**	80.9%	**99.0%**

Table 6.9. Per-object breakdown on the GRAB-Goal (cross-object) split.

We bound the velocity of our randomly generated trajectories to be between $[0, 2]$ m/s and bound the angles to be between $[0, 1]$ radian. With a probability of 0.2, a sharp turn could happen where the angle is between $[0, 2\pi]$. As the trajectories can not be too high or low, we bound the z-direction translation to be between $[0.1, 2.0]$. For orientation, we sample a random ending orientation at the end of the trajectory and interpolate it between the object's initial trajectory to obtain a sequence of target rotations.

6.6.8 Additional Ablations

In Table 6.8, we provide additional ablations left out due to space limitations. Comparing Row 1 (R1) and R4, we can see that on the GRAB dataset cross-object test set, a policy trained without the object shape latent code $\boldsymbol{\sigma}^{\text{obj}}$ can be on par with a policy with access to it. This is because the humanoid learned a general "grasping" for small objects, and the 5 testing objects do not deviate too much from these strategies. Also, upon inspection, R1 learns to rely on bi-manual manipulation and using two hands when it cannot pick it up with one hand, at which point the object shape no longer affects the grasping pose as much. As a result, R1 suffers a higher rotation error E_{rot}. On the GRAB cross-subject test (44 objects), R1 has a trajectory success rate of $\text{Succ}_{\text{traj}}$ 84.2%, worse than R4's 90.5%. R2 vs. R4 shows that the RNN policy is more effective than the MLP-based policy, confirming our intuition that some form of memory is beneficial for a sequential task, such as grasping and omnidirectional trajectory following. R3 studies the scenario where we provide ground truth full-body pose $\hat{\boldsymbol{q}}_t$ to the policy at all times, similar to the setting in PhysHOI [293] (though without the contact graph). Results show that this strategy leads to worse performance, and also prevents us from training on objects that do not have paired MoCap full-body motion. This indicates that the contact graph is needed to imitate human-object interaction precisely. Omnigrasp provides a flexible interface to support learning and testing on novel objects without needing paired ground-truth full-body motion.

6.6.9 Per-object Successrate breakdown

In Table 6.9, we break down the per-object success rate on the cross-object split of the GRAB dataset. Of the 5 novel objects, our model finds it hardest to pick up the toothpaste, which has an elongated surface. Upon inspection, we find that Omnigrasp will slip on the round edges of the toothpaste surface and fail to grasp the object. Compared to previous SOTA [26], Omnigrasp outperforms in all metrics and objects.

6.6.10 Additional Discussions

6.6.11 Alternatives to PULSE-X

One alternative way for reusing the motor skills from a motion imitator like PHC-X is to train a kinematic motion latent space to provide reference motion to drive PHC-X. Such a general-purpose kinematic latent space has been used in physics-based control for pose estimation [291] and animation [331]. However, few have been extended to include dexterous hands. These latent spaces, like HuMoR [231], model motion transition using an encoder $\boldsymbol{q}_\phi(\boldsymbol{z}_t|\hat{\boldsymbol{q}}_t, \hat{\boldsymbol{q}}_{t-1})$ and decoder $p_\theta(\hat{\boldsymbol{q}}_t|\boldsymbol{z}_t, \hat{\boldsymbol{q}}_{t-1})$ where $\hat{\boldsymbol{q}}_t$ is the pose at time step t and $\boldsymbol{z}_t$ is the latent code. $\boldsymbol{q}_\phi$ and $\boldsymbol{p}_\theta$ are trained using supervised learning. The issue with applying such a latent space to simulated humanoid control is twofold:

- The output $\hat{\boldsymbol{q}}_t$ of the VAE model, while representing natural human motion, does not model the PD-target (action) space required to maintain balance. This is shown in prior art [291,331], where an additional motion imitator is still needed to actuate the humanoid by imitating $\hat{\boldsymbol{q}}_t$ instead of using $\hat{\boldsymbol{q}}_t$ as policy output (PD-target).
- $\boldsymbol{q}_\phi$ and $\boldsymbol{p}_\theta$ are optimized using MoCap data, whose $\hat{\boldsymbol{q}}_t$ values are computed using ground truth motion and finite difference (for velocities). As a result, $\boldsymbol{q}_\phi$ and $\boldsymbol{p}_\theta$ handle noisy humanoid states from simulation poorly. Thus, [291] runs the kinematic latent space in an open-loop auto-regressive fashion without feedback from physics simulation (*e.g.* using $\hat{\boldsymbol{q}}_{t-1}$ from the previous time step's output rather than from simulation). The lack of feedback from physics simulation leads to floating and unnatural artifacts [291], and the imitator heavily relies on residual force control to maintain stability.

6.7 Limitations, Conclusions, and Future Work

Limitations. While Omnigrasp demonstrates the feasibility of controlling a simulated humanoid to grasp diverse objects and hold them to follow diverse trajectories, many limitations remain. For example, though the 6DoF input is provided in the input and reward, the rotation error remains to be further improved. Omnigrasp has yet to support precise in-hand manipulations. The success rate on trajectory following can be improved, as objects can be dropped or not picked up. Another area of improvement is to achieve *specific* types of grasps on the object, which may require additional input such as desired contact points and grasp. Human-level dexterity, even in simulation, remains challenging. For visualization of failure cases, see `supplement site`.

Conclusion and Future Work. In conclusion, we present Omnigrasp, a humanoid controller capable of grasping > 1200 objects and following trajectories while holding the object. It generalizes to unseen objects of similar sizes, utilizes bi-manual skills, and supports picking up larger objects. We demonstrate that by using a pretrained universal humanoid motion representation, grasping can be learned using simplistic reward and state designs. Future work includes improving trajectory following success rate, improving grasping diversity, and supporting more object categories. Also, improving upon the

humanoid motion representation is a promising direction. While we utilize a simple yet effective unified motion latent space, separating the motion representation for hands and body [15, 26] could lead to further improvements. Effective object representation is also an important future direction. How to formulate an object representation that does not rely on canonical object pose and generalizes to vision-based systems will be valuable to help the model generalize to more objects.

Chapter 7

Emergent Active Perception and Dexterity of Simulated Humanoids from Visual Reinforcement Learning

Figure 7.1. Perceptive Dexterous Control (PDC) enables a humanoid equipped with egocentric vision to search for, reach, grasp, and manipulate objects in cluttered kitchen scenes. We use visual perception as the sole interface for indicating which hand to use, which object to grasp, where to move, and which drawer to open.

7.1 Introduction

Human interaction with the world is fundamentally governed by active perception — a continuous process where vision, touch, and proprioception dynamically guide our movements and interactions [196, 197]. In this work, we study how a simulated humanoid, equipped only with egocentric perception, can perform tasks such as object transportation and articulated object manipulation in diverse household scenes. The combination of high-dimensional visual input and high-degree-of-freedom humanoid control makes this problem computationally demanding. As a result, humanoid controllers in animation and robotics often rely on privileged state information (*e.g.*, precise 3D object pose and shape), sidestepping the challenges of processing noisy visual input. This raises a key question: How can we effectively

close the egocentric perception-action loop to enable humanoids to interact with their environment using only their egocentric sensory capabilities?

s Perception-driven humanoid control presents unique challenges because it involves two computational intensive problems: dexterous whole-body control and visual perception in natural environments. While loco-manipulation demands precise coordination of balance, limb movements, and dexterity, incorporating vision introduces additional complexities of partial observability and the need for active information gathering. Due to these challenges, prior work [26,167,293] has relied mainly on privileged object-state information to achieve humanoid-object interaction, bypassing the difficulties of noisy sensory input and gaze control. While using privileged information provides a compact, pre-processed representation of the environment that includes objects outside the field of view, it fundamentally limits the emergence of human-like behaviors such as visual search since the humanoid has complete knowledge of object locations at all times. In contrast, learning from visual observations eliminates this limitation, but it also introduces several key challenges. The policy must learn to handle partial observability and tackle the increased input dimensionality (such as $128\times128\times3$ RGB images) — which is orders of magnitude larger than state representations. These challenges exacerbate the significant sample efficiency issues in reinforcement learning (RL) for whole-body humanoid control.

Another challenge in developing vision-driven humanoids is designing an input representation that facilitates learning multiple tasks simultaneously. Consider a kitchen cleaning scenario: the humanoid must identify specific objects in cluttered environments, understand task commands, and precisely place items in designated locations. Each of these steps requires clear task instructions. While natural language offers flexibility as an interface, it often lacks the precision required for exact object selection and placement in physical space. Previous approaches, such as Catch & Carry [184], have combined vision-based policies with phase variables, but this design limits scalability since each new task requires retraining the policy to handle new task variables. The ideal input representation should be task-agnostic, allowing the policy to learn in diverse manipulation scenarios without task-specific modifications.

To address these challenges, we introduce Perceptive Dexterous Control (PDC), a framework for learning human-like whole-body behaviors of simulated humanoids through perception-driven control. PDC controls humanoids to act solely from visual input (RGB, RGB-D, or Stereo) and proprioception. Rather than using state variables (*e.g.*, task phases, object information) to encode tasks, PDC specifies tasks directly through perception. Similarly to how augmented reality enhances human vision via informative markers and visual cues, we enhance the robot's vision using object masks and 3D markers to select objects and specify target locations. This vision-centric approach enables lifelong learning [254], allowing the agent to continuously adapt its skills to new tasks and complex scenes. To tackle the challenging whole-body dexterous control task, we leverage control priors learned from large-scale motion capture data. Combined with the natural household settings and dexterous hand tasks, these priors enable RL policies to develop emergent human-like behaviors — including active search and whole-body coordination.

To summarize, our contributions are as follows: (1) We demonstrate the feasibility of vision-driven, whole-body dexterous humanoid control in naturalistic household environments, a novel and challenging task that bridges perception and complex motor control; (2) We introduce a vision-driven task specification paradigm that uses visual cues for object selection, target placement, and skill specification, eliminating the need for predefined state variables; (3) We show that human-like behaviors, including active search and whole-body coordination, emerge from our vision-driven approach. Extensive evaluation shows how different visual modalities (RGB, RGBD, Stereo) and reward designs impact behaviors and task performance (*e.g.*, stereo outperforms RGB by 9% in success rate).

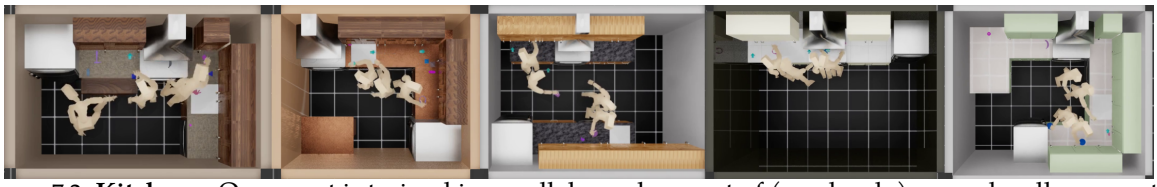


Figure 7.2. **Kitchens:** Our agent is trained in parallel on a large set of (randomly) procedurally generated kitchens. Each generated kitchen is structurally and visually different. Objects are spawned in random locations on the counter, and the agent starts in a random position and orientation within the scene. This diversity encourages the agent to learn general behaviors, such as search, and robust interaction.

7.2 Related Work

Visual Dexterous Manipulation. Learning visual dexterous policies for simulated [287, 312] and real-world [40, 42, 45, 149–151, 165, 220–222, 255, 256, 306] dexterous hands has gained increasing attention due to its relevance to real-world tasks such as object picking and manipulation. These approaches can be roughly divided into RL [149, 165, 256, 312] and behavior cloning [45, 151, 222] methods. PDC falls under RL, where the dexterous hands' behavior is learned through trial and error. Among RL methods, a popular paradigm is to first learn a state-space policy using privileged object information from simulation and then distill it into vision-based policies [165, 256, 287, 312]. However, to successfully distill, these methods assume a fixed camera pointing at a stationary table and do not involve active perception. For mobile and dexterous humanoids that loco-manipulate, where to look and how to pick up objects remain open questions.

Whole Body Loco-Manipulation. Controlling simulated [15,26,90,167,184,278,293,305,308,311] and real humanoids [97,163] to loco-manipulate requires precise coordination of the lower and upper body. Some prior work adopts a reference motion imitation approach [97,163,293,308], where the humanoid learns from reference human-object interactions. Others follow a generative approach [15,90,167,278,305,311], where the humanoid is provided high-level objectives (such as moving boxes [278], grasping objects [167], *etc*.) to complete but not a reference motion to follow. Closely related to ours, Omnigrasp [167] studies a state-space policy to grasp diverse objects using either hand (predominantly the right hand). HumanVLA [311] learns a language-conditioned visual policy for room arrangement via distillation. Compared to PDC, HumanVLA does not use dexterous hands and always assumes the object of interest and target location are within view of the humanoid during initialization. Similarly to our setting of egocentric vision, Catch & Carry [184] learns a vision-conditioned policy for carrying boxes and catching balls and uses phase variables and privileged information (object position for carrying). Compared to Catch & Carry, we involve dexterous hands, learn multiple tasks, and do not use privileged information.

Perception-as-Interface. The interface theorem of perception [105] argues that perception functions as an interface for guiding useful actions instead of reconstructing the 3D world. Recently, many [109, 193, 307] have explored using visual signals as indicators for policies to imbue [193] common sense or guide actions [307]. In games and augmented reality (AR), 3D markers and visual signals are also constantly used to indicate actions or give instructions. Building upon this idea, we leverage the redundant nature of images and use visual cues to replace state input for our RL policy.

Hierarchical Reinforcement Learning. The options framework, also known as skills [261], provides an abstraction layer for RL agents, enabling the separation between high-level planning and low-level execution [89, 165, 182, 211, 256, 274]. Reusable latent models have recently emerged as a promising hierarchical approach for humanoid control. First, a low-level latent-conditioned controller (LLC) is trained to generate a diverse repertoire of motions [170, 213, 275, 299]. Then, a high-level controller (HLC) is trained to utilize the LLC by predicting latents, effectively reusing the learned skills. This hierarchical

structure has been used in complex scenarios involving humanoid-object interactions [26, 167], where the separation between high-level task planning and low-level motion execution can significantly reduce the learning complexity.

Lifelong Learning. Interactive agents face the challenge of unpredictable operational environments. The framework of lifelong learning addresses this by continuously acquiring and transferring knowledge across multiple tasks sequentially over the agent's lifetime [254]. In reinforcement learning, lifelong learning approaches have demonstrated superior performance in complex tasks through knowledge reuse and retention [6, 180, 194, 274]. Our work encodes all tasks via visual cues, enabling the policy to learn new tasks through fine-tuning without architectural modifications. We show that this capability is crucial and allows the PDC to master simple skills before adapting to diverse, complex scenes.

7.3 Method

To tackle the problem of visual dexterous humanoid control, we train a policy to output joint actuation based on current visual and proprioceptive observations. In Section 7.3.1, we define the task settings of our visual dexterous control problem. In Section 7.3.2, we describe our perception-as-interface design to enable vision-only task learning. Finally, in Section 7.3.3, we detail how we learn our control policy.

7.3.1 Task Definitions

We study two scenarios for visual dexterous whole-body control: (1) a tabletop setting where the humanoid manipulates objects on an isolated table floating in midair, and (2) a naturalistic kitchen environment with diverse objects, randomized placements, and articulated cabinets, as visualized in fig. 7.2. The tabletop scenario, inspired by Omnigrasp [167], is devoid of complex visual backgrounds, providing a controlled training environment for visual-dexterous grasp learning. Here, the humanoid is initialized facing the table with one object on the table. The policy is trained to approach the table, pick up an object, move it to a designated 3D location, and place it down. The kitchen scene presents a significantly more challenging setting, featuring a photorealistic background, distractor objects, and articulated drawers. In this environment, we consider both object transport and drawer manipulation. In the object transport task, the object position is unknown, and the humanoid must first locate the target object, pick it up, transport it to the specified 3D location, and release it. In the articulated cabinets task, the agent is tasked with opening and closing specified cabinets on command.

Scene Generation. To create diverse environments, we leverage SceneSynthesizer [71] to generate six distinct kitchen types (galley, flat, L-shaped, with island, U-shaped, and peninsula). For each type, we produce 80 randomized configurations with variations in both structural elements (*e.g.*, cabinet quantity and placement) and surface textures. We pre-compute 20 valid humanoid spawning positions and 64 object placement locations within each generated scene, ensuring all elements are positioned without geometric intersections or scene penetrations. We create 512 environments for training and 64 for testing.

7.3.2 The Visual Perception Interface

To tackle the task of object search, grasp, goal reaching, release, and drawer opening, prior art resorts to phase variables [184] or modular state machines [123] to decide which part of the task the policy should execute on. While adding phase variables, object information, and task instructions such as 3D goal coordinates is helpful for state-space policy training, it limits the possibility of learning new tasks. Any change in the task requires re-training the policy to accommodate new state variables. We propose to use perception as the sole interface for task specification. Leveraging vision as a unified interface produces

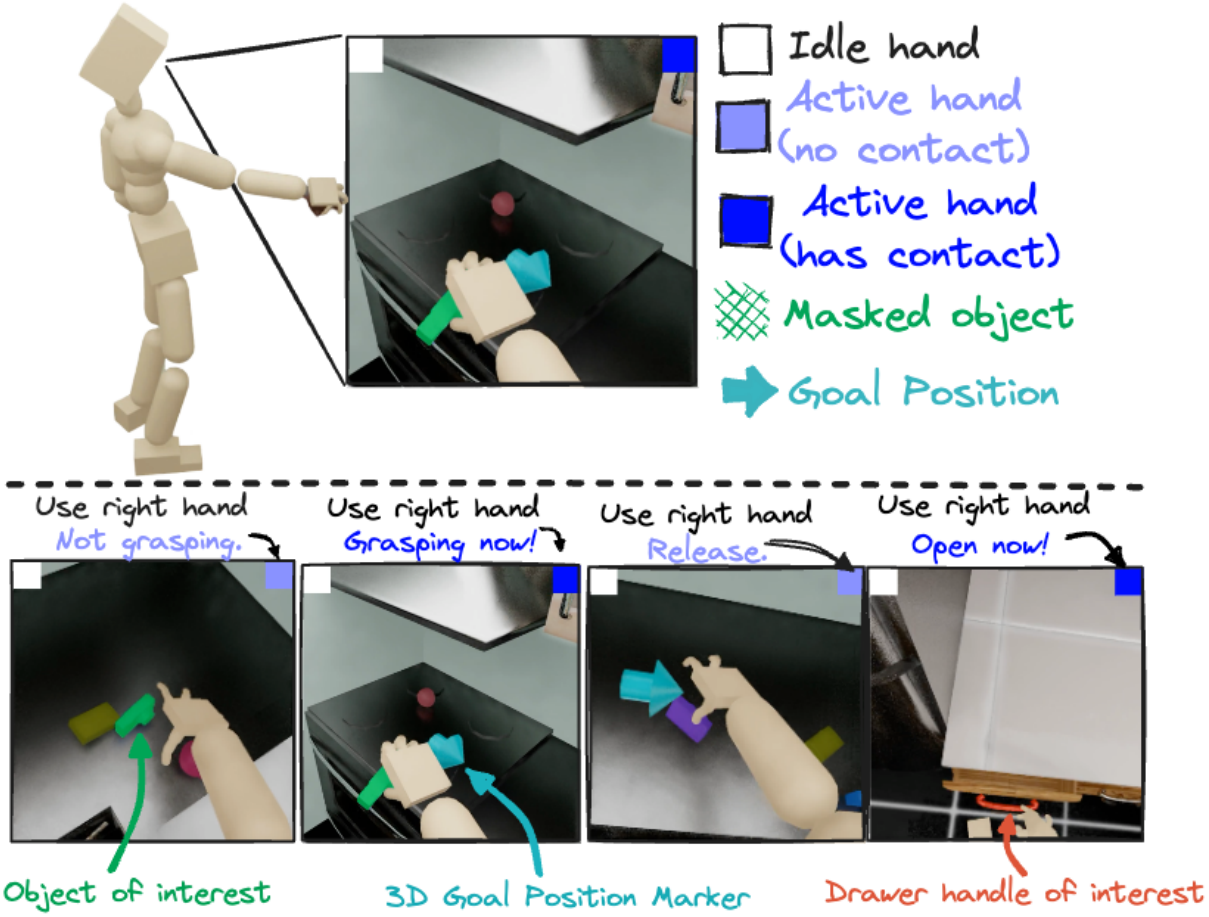


Figure 7.3. **Perception-as-Interface:** PDC instructs the policy through visual signals. We overlay the object of interest using a green mask, use a 3D blue marker to indicate target location, and use colored 2D squares (top corners) to inform the agent which hand to use and when to grasp and release.

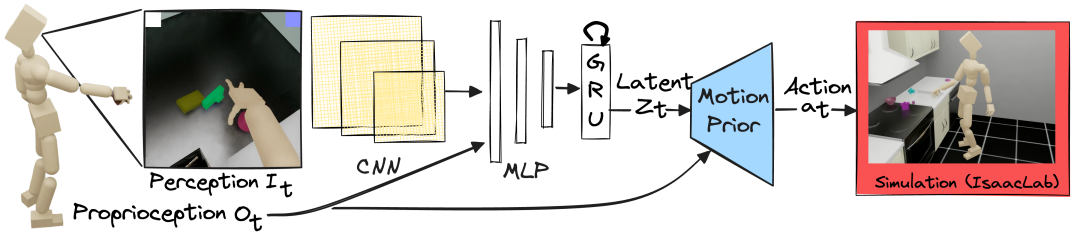


Figure 7.4. We use perception and proprioception as input to the network, processed by a simple CNN-GRU-MLP architecture.

a life-long learning agent that can adapt (through fine-tuning) to new tasks without any architectural changes. Sometimes, visual signals could even be floating text for precise task instructions. Our approach embraces this principle, using visual indicators to specify tasks, as illustrated in fig. 7.3.

Object Selection. The kitchen scenes contain multiple objects, making object specification a significant challenge, particularly when precise location information is unavailable. For instance, selecting a particular apple from a cluster becomes ambiguous without explicit spatial coordinates. To address this challenge, we leverage 2D semantic segmentation [130] as a visual selection mechanism. Our approach applies alpha blending using the segmentation mask to highlight the target object with a distinctive bright green overlay directly **in the image space**. This visual differentiation enables the agent to identify the correct object and transform the object selection problem into a visually guided task. To differentiate between object grasping and drawer opening, we paint drawers handles bright red.

Object Reaching Goals. Once the object is picked up, the next step is specifying its destination. While 3D target locations can specify goals, such input will become redundant for tasks like object search or cabinet opening and lacks adaptability to new scenarios. To overcome these limitations, we instead *render* a 3D arrow directly in the agent's visual field to indicate the desired destination. This interface resembles information markers commonly used in 3D games or AR applications and provides an intuitive and visually grounded objective for manipulation tasks.

Handedness, Pickup Time, and Release Time. We train our agent for bimanual control, enabling it to use the left hand, right hand, or both hands on command. Additionally, the humanoid must be able to pick up and place objects as instructed. To achieve this, we color the visual field's upper right and left corners to indicate which hand should grasp the object. A small colored block indicates hand usage: white signals the hand should remain idle, purple designates a hand that should be used for contact but should not currently be in contact, and blue signifies active contact now (grasping). The purple signal guides the humanoid in determining which hand to use to reach out to the object.

We believe that perception-as-interface [105] offers near-infinite extensibility for vision-based agents, with an extreme case being using floating text to convey instructions.

7.3.3 Learning Perceptive Dexterous Control

We formulate our task using the general framework of goal-conditioned RL. The learning task is formulated as a Partially Observed Markov Decision Process [13, POMDP] defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$ of states, observations, actions, transition dynamics, reward function, and discount factor. The simulation determines the state $\boldsymbol{s}_t \in \mathcal{S}$ and transition dynamics $\mathcal{T}$, and a policy computes the action $\boldsymbol{a}_t \in \mathcal{A}$ based on the partial observation $\boldsymbol{o}_t \in \mathcal{O}$. The observation $\boldsymbol{o}_t \in \mathcal{O}$ is a partial representation of the full simulation state $\boldsymbol{s}_t$, combined with the task instructions. We train a control policy $\pi_{\text{PDC}}(\boldsymbol{a}_t|\boldsymbol{o}_t)$ to compute joint actuation $\boldsymbol{a}_t$ from observations. The reward function $r_t = \boldsymbol{\mathcal{R}}(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$ is defined using the full simulation state $\boldsymbol{s}_t$ to guide learning. The policy optimizes the expected discounted reward $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$ and is trained using proximal policy optimization (PPO) [245].

Observation. The policy observes the proprioception (the observation of oneself) and camera image $\boldsymbol{o}_t \triangleq (\boldsymbol{o}_t^{\text{P}}, \boldsymbol{I}_t)$. The proprioception $\boldsymbol{o}_t^{\text{P}} \triangleq (\boldsymbol{\theta}_t, \boldsymbol{p}_t, \boldsymbol{v}_t, \boldsymbol{\omega}_t, \boldsymbol{c}_t)$ consists of the 3D joint rotation $\boldsymbol{\theta}_t \in \mathbb{R}^{J\times 3}$, position $\boldsymbol{p}_t \in \mathbb{R}^{J\times 3}$, linear velocity $\boldsymbol{v}_t \in \mathbb{R}^{J\times 3}$, angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^{J\times 3}$, and the contact forces on each hand $\boldsymbol{c}_t \in \mathbb{R}^{J_H\times 3}$. Due to computational challenges posed by larger images, prior work focused on low-resolution inputs (*e.g.*, 40×30×3 [278]); our experiments demonstrate the performance benefits of higher-resolution visual inputs. We evaluate RGB (128×128×3), RGBD (100×100×4), and stereo (2×80×80×3), with the latter two using slightly smaller dimensions due to GPU memory constraints.

Reward. To train PDC, we provide it with dense rewards that help and guide its behavior, as RL agents struggle when trained with sparse objectives (*e.g.*, $r_t = \mathbf{1}_{\text{object on marker}}$). For the grasping task, we de-

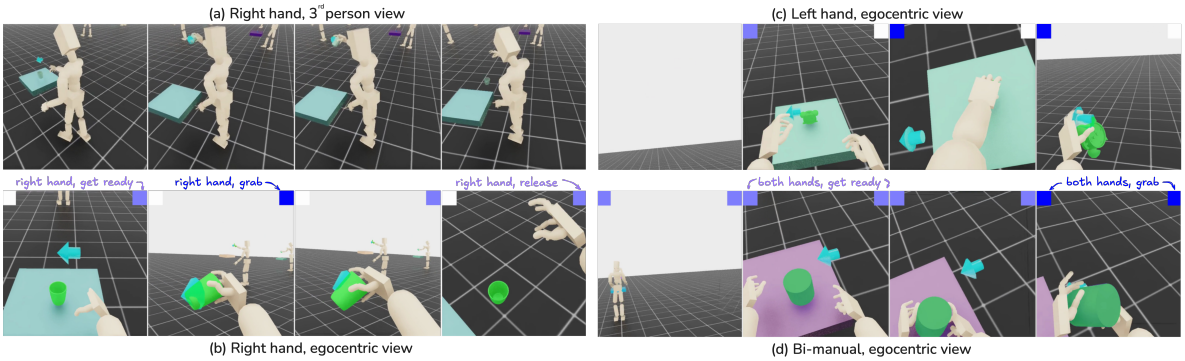


Figure 7.5. **Tabletop:** PDC is instructed directly from visual signals. A visual (top left and/or right) indicates in purple whether the corresponding hand should be prepared for contact. Changing to dark-blue indicates that contact should be made. Instructing the agent to use both hands enables it to transport larger objects. Changing the indicator back to purple instructs the agent to release the object.

signed two rewards, a reward aimed to measure grasp and object pose, and another (optional) to guide the gaze. The grasp reward is:

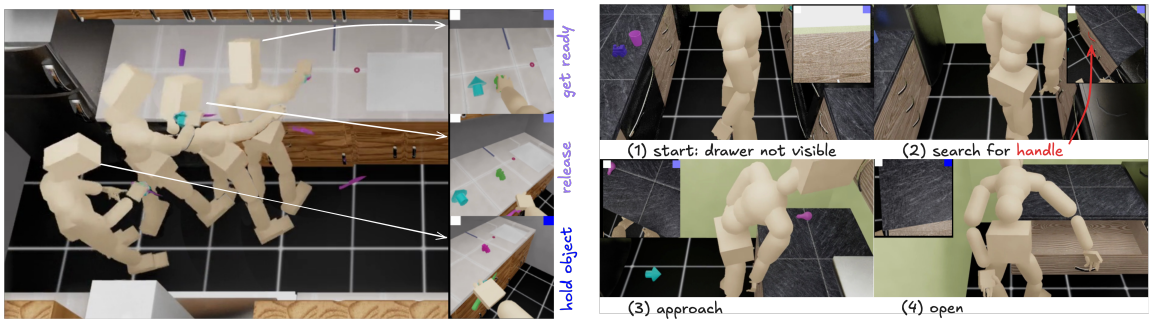
$$r_t^{\text{PDC}} = \begin{cases} r_t^{\text{approach}} + r_t^{\text{lookat}}, & \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 > 0.2 \text{ and } t < \lambda_{\text{start}} \\ r_t^{\text{pre-grasp}} + r_t^{\text{lookat}}, & \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 \leq 0.2 \text{ and } t < \lambda_{\text{start}} \\ r_t^{\text{obj}} + r_t^{\text{lookat}}, & \lambda_{\text{start}} \leq t < \lambda_{\text{end}} \\ (1 - \mathbf{1}_{\text{has-contact}}), & t \geq \lambda_{\text{end}} \end{cases} \tag{7.1}$$

This reward is split into 4 sequential segments: a time scheduler determines whether the agent should be in the approach, grasp, or release phase. The pre-grasp phase is broken into 2 segments, reach and pre-grasp. The pre-grasp [61, 167] is provided when the hand is $< 0.2m$ from the target object. This encourages the hand to match a pre-computed pre-grasp $\hat{\boldsymbol{q}}^{\text{pre-grasp}} \triangleq (\hat{\boldsymbol{p}}^{\text{pre-grasp}}, \hat{\boldsymbol{\theta}}^{\text{pre-grasp}})$, a single hand pose consisting of the hand translation $\hat{\boldsymbol{p}}^{\text{pre-grasp}}$ and rotation $\hat{\boldsymbol{\theta}}^{\text{pre-grasp}}$. If the object is successfully grasped, we switch to the object 3D location reward r_t^{obj} to encourage transporting the object to specific locations. After the grasping should end ($t \geq \lambda_{\text{end}}$), we encourage the agent to release the object: $\mathbf{1}_{\text{has-contact}}$ determines if any hand has contact with the object. The optional look-at reward

$$r_t^{\text{lookat}} = \begin{cases} r_t^{\text{lookat-object}}, & \text{and } t < \lambda_{\text{start}} \\ r_t^{\text{lookat-marker}}, & \text{and } t \geq \lambda_{\text{start}} \end{cases}, \tag{7.2}$$

is also time-conditioned. First, the humanoid is rewarded for looking at the object. After the contact start time, the humanoid is encouraged to look at the 3D target marker. The look-at reward is computed as $r_{\text{lookat}} = 1 - \sqrt{1 - (\mathbf{v}_{\text{gaze}} \cdot \mathbf{v}_{\text{eye-target}})}$, where $\mathbf{v}_{\text{gaze}}$ is the gaze vector and $\mathbf{v}_{\text{eye-target}}$ is the eye to target (object or marker) vector. While we encourage the policy to always look in the object's direction, search (looking left and right) emerges when the object is not always initialized in the view. To train the drawer open policy, we use the same pregrasp and approach reward, and change the object reward position r_t^{obj} to drawer opening reward r_t^{drawer}. For details about each reward term, please refer to Section 7.6.9

Early Termination. Equally important to reward design, early termination [208] also plays an important role in shaping the agent's behavior as it provides a strong negative reward signal to the agent. We early terminate the training episode in the following scenarios: (1) the agent is not in contact with the object by λ_{start} grasping time, (2) the agent is still in contact of the object after contact end time λ_{end}, and (3) if the target object is more than 0.25m away from the marker for more than 2 seconds.



(a) Multi-object manipulation. (b) Search and open drawers.

Figure 7.6. **Kitchen task:** Here, the agent must master multiple skills. First, the agent must search and find the objective. A target object is marked in green, whereas a target drawer handle in red. Once found, the agent must approach the target before it can interact with it.

Policy Architecture. We use a simple CNN-GRU-MLP architecture for the policy, illustrated in Figure 7.4. The CNN processes raw image inputs, extracting high-level spatial features. The GRU provides recurrent memory, enabling the agent to handle temporal dependencies of this partially observable task. Due to the large throughput of data when training with an image-conditioned RL policy, we use a simple two-layer CNN architecture that has been used in sim-to-real robotics settings [149]. For RGBD, the depth channel is treated as an additional image channel. For stereo, we use a Siamese network [133] and process the two images with the same CNN and concatenate the features. We also experiment with frozen pretrained visual encoders like ResNet [95] and ViT [65], though they do not yield boosts.

Humanoid Motion Representation. Due to the difficulty of controlling a high-dimensional humanoid with dexterous hands, many prior works resort to reference motion imitation [293, 308] or overfitting to a single type of action [15]. We do not use any reference motion and utilize the recently proposed PULSE-X [167, 170] as the reusable, general-purpose, low-level controller. PULSE-X is a conditional VAE (cVAE) that is trained to reproduce motions from the AMASS MoCap dataset [176]. At a high level, PULSE-X learns a proprioception-conditioned latent space that can be decoded into human-like actions $\mathcal{D}_{\text{PULSE-X}}(\boldsymbol{a}_t|\boldsymbol{z}_t^{\text{omnigrasp}}, \boldsymbol{o}_t^{\text{p}})$. Using PULSE-X, we define the action space of our policy with respect to the prior $\boldsymbol{a}_t = \mathcal{D}_{\text{PULSE-X}}(\pi_{\text{PDC}}(\boldsymbol{z}_t^{\text{omnigrasp}}|\boldsymbol{o}_t^{\text{p}}, \boldsymbol{I}_t), \boldsymbol{o}_t^{\text{p}})$ and train it using hierarchical RL. For more information about PULSE-X, please refer to Section 7.6.8.

Tabletop training: the humanoid is initialized at a random location facing (but not looking at) the object. The target goal position is sampled randomly above the tabletop and changes periodically. **Kitchen training:** the humanoid is initialized randomly in an empty space facing a random direction. Target object positions are sampled randomly on the support surface and changed periodically (see samples in `supplement site`). **Optimization:** Unlike imitation-based methods [293, 298, 308], we do not use any reference human motions during training. We follow the standard on-policy RL [245, PPO] training procedure, interleaving sample collection and policy updates. To successfully solve the kitchen scene, we warm-start the policy using the trained table-top policy. *This is enabled* by our perception-as-interface paradigm, as we can reuse the grasping capability from the tabletop agent. For the kitchen scene, half of the environments learn the grasping task while the other half opens drawers.

7.4 Experiments

Baselines. As there are few prior attempts at visual dexterous whole-body control, we compare against a

		GRAB-Train (25 seen objects)								GRAB-Test (5 unseen objects))							
Method	Vision	Succ ↑	E_{pos} ↓	$\text{Succ}_{\text{right}}$ ↑	E_{pos} ↓	$\text{Succ}_{\text{left}}$ ↑	E_{pos} ↓	Succ_{bi} ↑	E_{pos} ↓	Succ ↑	E_{pos} ↓	$\text{Succ}_{\text{right}}$ ↑	E_{pos} ↓	$\text{Succ}_{\text{left}}$ ↑	E_{pos} ↓	Succ_{bi} ↑	E_{pos} ↓
Omnigrasp	✗	99.1%	8.9	99.1%	8.9	98.7%	13.0	99.7%	12.5	70.6%	11.2	70.4%	11.2	-	-	100.0%	18.7
PDC-RGB	✓	87.5%	74.4	88.7%	72.6	83.0%	77.4	78.7%	98.3	90.1%	100.5	90.0%	101.0	-	-	**100.0%**	98 39.1
PDC-RGBD	✓	86.9%	61.5	88.1%	60.7	80.1%	**68.6**	83.9%	**59.5**	85.2%	74.6	85.1%	74.8	-	-	**100.0%**	**41.8**
PDC-stereo	✓	**96.4%**	**51.9**	**96.9%**	**47.7**	**92.9%**	79.2	**87.7%**	57.5	**91.8%**	**61.0**	85.8%	61.1	-	-	**100.0%**	41.6

Table 7.1. Quantitative results on visual object grasps and target goal reaching on the GRAB dataset. The training set contains 82.4% right hand, 12.1% left hand, and 5.4% both hands. The test set contains 99% right hand grasp and 1% both hands.

	GRAB-Train, Scene-Train				GRAB-Test, Scene-Train				GRAB-Test, Scene-Test				Drawer, Scene-Train		Drawer, Scene-Test	
	Search	Pick up	Goal Reaching		Search	Pick up	Goal Reaching		Search	Pick up	Goal Reaching		Search	Drawer Open	Search	Drawer Open
Method	$\text{Succ}_{\text{search}}$ ↑	$\text{Succ}_{\text{grasp}}$ ↑	$\text{Succ}_{\text{traj}}$ ↑	E_{pos} ↑	$\text{Succ}_{\text{search}}$ ↑	$\text{Succ}_{\text{grasp}}$ ↑	$\text{Succ}_{\text{traj}}$ ↑	E_{pos} ↑	$\text{Succ}_{\text{search}}$ ↑	$\text{Succ}_{\text{grasp}}$ ↑	$\text{Succ}_{\text{traj}}$ ↑	E_{pos} ↑	$\text{Succ}_{\text{search}}$ ↑	$\text{Succ}_{\text{drawer}}$ ↑	$\text{Succ}_{\text{search}}$ ↑	$\text{Succ}_{\text{drawer}}$ ↑
PDC- Stereo	98.1%	85.1%	65.5%	162.2	98.9%	79.1 %	52.8%	147.6	95.7%	80.2	53.6%	154.4	98.8%	63.7%	98.0%	64.0%

Table 7.2. Quantitative results on visual object grasps, target goal reaching, and drawer opening in the kitchen scene. We test on different combination of train/test scenes and train/test objects.

modified version of a SOTA state-space policy, Omnigrasp [167]. The state-space policy always observes the object and target location and serves as the oracle policy. We extend the original Omnigrasp policy to support hand specification and placing down the object. The state-space policy is trained using the exact same reward as the visual controllers, including the look-at reward. Please refer to Section 7.6.7 for more information on this baseline.

Implementation Details. Simulation is conducted in NVIDIA IsaacLab [185] and rendered using tiled rendering. Due to the significantly increased data throughput for training visual RL, we simulate 192 environments per-GPU across 8 Nvidia L40 GPUs in parallel. Simulation is conducted in 120Hz while the control policy runs at 30Hz. The tabletop policy training takes around 5 days to converge. Fine-tuning a trained tabletop policy for the kitchen scene requires an additional 5 days. This results in approximately 5×10^9 samples, roughly 4 years' worth of experience. Our simulated humanoid follows the kinematic structure of SMPL-X [204] using the mean body shape [167, 169, 172]. This humanoid has $J = 52$ joints, of which 51 are actuated. Between these joints, 21 joints belong to the body, and the remaining $J_H = 30$ joints are for two hands. All joints are 3 degrees-of-freedom (DoF) spherical joints actuated with PD controllers, resulting in $\boldsymbol{a}_t \in \mathbb{R}^{51\times 3}$. To mimic human perception, we attach camera(s) to the head of the humanoid, roughly at the position of the eyes. For stereo, we place the cameras 6 cm apart, similar to human eye placements.

Object Setups. We use a subset of the GRAB [266] dataset (25 out of 45 objects) that contains household objects for training and keep the testing set (5 objects) for both the kitchen and tabletop setting. We obtain the pre-grasp for the reward and hand specification from the MoCap recordings provided by the GRAB dataset: *e.g.* if the pre-grasp uses both hands, the episode will use the corresponding hands. We also provide qualitative results on larger and more diverse objects on the OMOMO [142] and Oakink [316] datasets (see `supplement site`).

						GRAB-Test (5 unseen objects))					
idx	Feature Extractor	Resolution	Input Modality	Lookat Reward	Distillation	$\text{Succ}_{\text{grasp}}$ ↑	E_{pos} ↓	$\text{Succ}_{\text{right}}$ ↑	E_{pos} ↓	Succ_{bi} ↑	E_{pos} ↓
1	ViT	128	RGB	✓	✗	61.4%	142.9	61.4%	142.8	60.0%	161.3
2	ResNet	128	RGB	✓	✗	68.5%	194.7%	68.7%	194.0	40.0	375.7
3	CNN	128	RGB	✓	✓	68.2%	**16.2**	68.6	**16.3**	10.0%	79.0
4	CNN	32	RGB	✓	✗	80.7%	116.4	80.6%	115.7	**100.0%**	189.9
5	CNN	64	RGB	✓	✗	80.2%	100.1	80.1%	100.2	90.0%	82.0
6	CNN	128	RGB	✗	✗	89.6%	85.6	89.5%	85.9	100.0%	**48.0**
7	CNN	128	RGB	✓	✗	**90.1%**	100.5	**90.0%**	101.0	**100.0%**	**39.1**

Table 7.3. Ablation on various strategies of training PDC.

7.4.1 Results

As motion is best seen in videos, we strongly recommend that readers view the qualitative videos.

Metrics. In the tabletop scenario, the agent is evaluated on its ability to pick up an object, track a marker located 30cm above the table, and then release the object. An episode is considered successful if the humanoid picks up the object using the correct hand, reaches the target location (<25cm), and then places the object back down. The sequence of actions is time-conditioned and specified through visual cues. The policy has 2 seconds to pick up the object and 2 seconds to reach the target. We report the distance of the object from the visual marker E_{pos}, after the object is picked up. We also provide a breakdown of the success rate based on which hand to use — right $\text{Succ}_{\text{right}}$, left $\text{Succ}_{\text{left}}$, and both hands Succ_{bi}. The kitchen scene presents an additional challenge. The policy must search for the object, reach and pick it up, and then follow the marker. Here, the marker moves across the counter (instead of straight lifting). We report the search performance $\text{Succ}_{\text{search}}$, measuring whether the object comes into view; the grasp success $\text{Succ}_{\text{grasp}}$, measuring whether the correct object is grasped; and trajectory following success $\text{Succ}_{\text{traj}}$, which shows whether the object follows the trajectory (<25cm) and is successfully released at the end. Finally, we measure the policy's ability to open drawers $\text{Succ}_{\text{drawer}}$. For the kitchen setup, we test in both training scenes and *unseen scenes*.

Tabletop: Object Lifting. We report the results in Table 7.1 and Figure 7.5. The oracle state-space policy reaches a high success rate for the training set, yet its success rate drops on the test set. On the other hand, the visual policy reaches a similar success rate across both train and test. This result shows that while the ground-truth object shape information used by the state-space policy (extracted using BPS [219]) can help achieve high success rate during training, vision provides better generalization capabilities. Since most of the reference pre-grasps in the data use the right hand, PDC performs better at using the right hand than using the left hand and both hands. In terms of vision-format, stereo outperforms both RGBD and RGB, with RGBD coming in second. Surprisingly, stereo outperforms RGBD across all metrics, suggesting that stereo depth estimation emerges from the task of grasping and reaching target goals.

Kitchen Scenes: Search, Grasp, Move, and Drawers. For the kitchen scene, as shown in Table 7.2 and Figure 7.6, our policy achieves a high search and grasping success rate, even when the object is deep into the counter and the humanoid is required to lean on the counter to reach the object. The goal reaching success rate is lower compared to the tabletop setting, partially due to the harder goal positions in the kitchen setting (random positions above the counter). PDC is robust across unseen objects and kitchen scenes without large performance degradation on unseen objects and scenes. Finally, the drawer-opening task achieves a high success rate in finding and opening the drawer.

7.5 Ablation and Analysis

Pretrained Vision Encoders. In Section 7.4 Row 1 (R1), R2, and R7, we study using visual encoders frozen pretrained using ImageNet [63]. The results show that although the policy learns to pick up using these pretrained features, it does not reach comparable success rates to learning from scratch. We hypothesize that the features trained from ImageNet are insufficient to close the perception-action loop, and training from scratch or fine-tuning the visual feature-extractor is needed to learn better visual features.

Distillation vs From Scratch. Comparing R3 and R7, we can see that distillation achieves a much lower success rate compared to training from scratch. Distillation does reach a better position error, indicating that predicting the visual marker position in 3D can be a good auxiliary loss.

Resolution. Comparing R4, R5, and R7: the policy success rate increases as the resolution of the images increases. This shows that higher resolution does have benefits in terms of grasp success and goal

reaching. Notice that the stereo-model using two 80x80 resolution cameras achieves the highest success rate—achieving the highest success rate within our computational budget.

Look-at Reward. R6 vs R7 shows that without the look-at reward, the policy is still able to learn the table-top policy and achieves comparable results. However, the look-at reward helps shape the search behavior for the kitchen scene.

Analysis: Emergent Search Behavior. For the kitchen scene, our policy learns interesting emergent behavior due to the setup of the task. As can be seen from the `supplement videos`, the policy learns to look left & right and scans the room (sometimes turns 360) while searching for the object. Also, we observe that the humanoid learns to scan the counter to find the object, a behavior that emerges from our object spawn locations. Also, the marker is not guaranteed to be in view when the object has been grasped. We observe that the humanoid learns to *search left & right* for the marker. Such behavior emerges from the reward to look at the object and the complex kitchen setting and is not observed in the simpler tabletop setting.

Analysis: Object Selection. Our masking-as-object-selection design enables PDC to grasp objects of interest, even when objects of the same shape and geometry are in the view. In Table 7.2, results are reported when 6 objects are together in the scene. These results show high search and pickup rates, showing that PDC can find the right object.

Analysis: Multi-Task Learning. Our kitchen policy is warm-started from our trained tabletop policy. While the trained-from-scratch policy does learn to search (achieving a 64.3% success rate in search) for the object, it does not succeed in picking up the objects. Our perception-as-interface enables continuously adapting the policy to learn in different scenarios (table top $\rightarrow$ kitchen) and different tasks (object transportation and drawer opening).

7.6 Additional Details

7.6.1 Supplementary Website

As the resulting behaviors are hard to convey through text and images, we also provide `supplement videos`.

7.6.2 Teaser

The teaser video on the top illustrates the diverse environments in which PDC is trained. Across 6 different types of kitchens, spanning 80 random configurations and textures each. This diversity enables our agent to learn general behaviors, resulting in emergent search and dexterity.

7.6.3 Tabletop

The tabletop videos showcase the 3 control schemes. We provide a 3rd person video in addition to the agent's own visual (rendered at a higher quality for ease of viewing). The visual markers in the top corners of the agent's view indicate what each hand should do — stay idle, be ready to engage, engage, and release. We show examples of right hand, left hand, and also both-hand manipulation.

This is not limited to seen objects. Our PDC agent generalizes to new and unseen objects, directly from vision.

GRAB-Train (25 Seen Objects)										
Object	bowl	hammer	flashlight	mouse	duck	wineglass	scissors	airplane	stapler	torusmedium
Success Rate	100%	98.5%	95.8%	94.6%	90.4%	96.7%	89.1%	87.8%	95.5%	100%
Object	banana	cylindersmall	waterbottle	watch	stamp	alarmclock	headphones	phone	cylindermedium	flute
Success Rate	100%	100%	100%	99.0%	100%	95.6%	97.1%	93.2%	98.3%	95.8%
Object	cup	fryingpan	lightbulb	toothbrush	knife					
Success Rate	92.6%	98.1%	100%	98.0%	93.0%					
GRAB-Test (5 Unseen Objects)										
Object	apple	binoculars	camera	mug	toothpaste					
Success Rate	97.4%	65.5%	99.3%	93.3%	95.7%					

Table 7.4. Per-object success breakdown for the PDC-stereo policy in the tabletop setting.

7.6.4 Kitchens

Object transport

In the kitchen demonstrations, we first showcase the task of object transport. Here, the agent needs to identify the target object. There are 5 distractors (objects that are not the target). The target object is marked using a green mask overlay.

Once the object is picked up, the agent needs to find the target marker and bring the object towards it.

Unlike the tabletop which mainly focuses on lifting the object, here the agent needs to transport the object to a new location in the kitchen. Every 300 frames (10 seconds) we instruct it to release the object and randomly sample a new target object.

These scenes were not observed during training, showcasing the abilities of our agent to generalize. It exhibits traits such as search, where it scans the scene for the object. Moreover, an interesting emergent property is that it scans the counter-tops, as it has learned that is where kitchen objects are located (in our tasks).

Articulated drawers

Following the object transport, we showcase the ability of the agent to open drawers. These are highlighted in the agents point of view using a red overlay. The agent learns to insert its fingers through the handle in order to obtain a better grip. It then leans back with its body in order to pull the drawer open.

7.6.5 Failure cases

Finally we showcase some common failure cases. For example, although our interface enables selecting which hand to utilize, it requires a system to determine the hand-to-use apriori. When an object can not be picked up with the selected hand, the agent will fail. In these videos we show what happens when a single hand is selected to pick up a large object. The same objects are successfully picked up in the tabletop examples above, using both hands, yet fails when attempted with a single hand.

7.6.6 Implementation Details

7.6.7 State-space Policy

We learn a state-space policy similar to Omnigrasp [167] but with the modifications that instead of using desired object trajectory, we only provide a single target position. Also, Omnigrasp does not have the ability to specify which hand to use, nor can it put the object down on command. To enable these capabilities, we add additional phase variables to inform the policy of the current command. Specifically, the

GRAB-Train (25 Seen Objects)										
Object	bowl	hammer	flashlight	mouse	duck	wineglass	scissors	airplane	stapler	torusmedium
Success Rate	50.0%	62.5%	72.0%	74.5%	87.0%	77.0%	65.0%	60.5%	70.5%	80.5%
Object	banana	cylindersmall	waterbottle	watch	stamp	alarmclock	headphones	phone	cylindermedium	flute
Success Rate	76.0%	56.0%	60.5%	69.5%	63.5%	58.5%	64.0%	66.5%	80.5%	74.5%
Object	cup	fryingpan	lightbulb	toothbrush	knife					
Success Rate	66.0%	43.5%	77.5%	42.0%	39.5%					
GRAB-Test (5 Unseen Objects)										
Object	apple	binoculars	camera	mug	toothpaste					
Success Rate	63.0%	31.0%	62.0%	53.0%	59.0%					

Table 7.5. Per-object success breakdown for the PDC-stereo policy in the kitchen setting.

	AMASS-Train				
Method	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
PHC-X – IsaacGym	99.9 %	29.4	31.0	4.1	5.1
PULSE-X – IsaacGym	99.5 %	42.9	46.4	4.6	6.7
PHC-X – IsaacLab	99.9 %	25.4	26.7	4.8	5.2
PULSE-X – IsaacLab	99.6 %	29.1	30.1	4.2	5.1

Table 7.6. Imitation result on AMASS (14889 sequences).

observation input for our omnigrasp policy is

$$\boldsymbol{o}_t^{\text{Omnigrasp}} \triangleq (\boldsymbol{s}_t^{\text{p}}, \boldsymbol{p}_t^{\text{obj}}, \boldsymbol{\theta}_t^{\text{obj}}, \boldsymbol{\sigma}^{\text{obj}}, \hat{\boldsymbol{p}}_{t+1}^{\text{obj}}, \boldsymbol{h}_t),$$

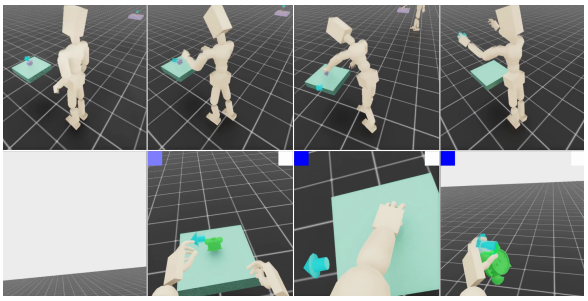
where $\boldsymbol{p}_t^{\text{obj}} \in \mathcal{R}^3$ object translation, $\boldsymbol{\theta}_t^{\text{obj}} \in \mathcal{R}^6$ object rotation, and object shape latent code $\boldsymbol{\sigma}^{\text{obj}} \in \mathcal{R}^{512}$ are privileged information provided by simulation. We compute the shape latent code $\boldsymbol{\sigma}^{\text{obj}}$ using BPS [219] with 512 randomly sampled points. The target object position $\hat{\boldsymbol{p}}_{t+1}^{\text{obj}} \in \mathcal{R}^3$ specifies the 3D position of where the object's centroid should be, and hand information $\boldsymbol{h}_t \in \mathcal{R}^{512}$ provides information about which hand to use and when to grasp and put down. $\boldsymbol{h}_t \triangleq (\boldsymbol{h}_t^{\text{left}}, \boldsymbol{h}_t^{\text{right}}, \boldsymbol{h}_t^{\text{time}})$ contains the indicator variables to indicate whether to use the left $\boldsymbol{h}_t^{\text{left}} \in \mathcal{R}^1$ or the right $\boldsymbol{h}_t^{\text{right}} \in \mathcal{R}^1$ hand. The time variable $\boldsymbol{h}_t^{\text{time}} \in \mathbb{R}^{10}$ encodes whether the policy should grasp or release the object within the next second, sampled at 0.1s intervals.

We train the state-space policy using the same training procedure as our vision-guided policy and use the same architecture networks (removing the visual encoder). The humanoid motion prior is also the same as the vision-conditioned policy. The state-space policy is trained using a similar number of samples as the visual policy via multi-GPU RL training.

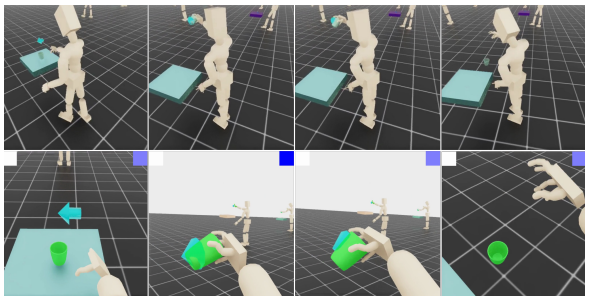
7.6.8 Humanoid Motion Prior

We use a similar training procedure to obtain our humanoid motion representation (PULSE-X) as Omnigrasp [167]. First, we train a motion imitator, PHC-X [169], on the cleaned AMASS dataset augmented with hand motion. Then, we distill the PHC-X policy into the PULSE-X CVAE using DAgger [239].

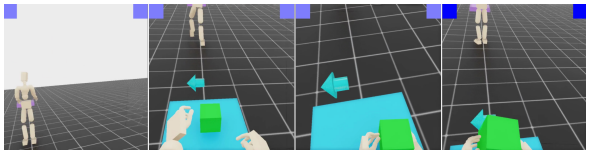
Motion Imitator, PHC-X. The task of motion imitation involves training a control policy to follow per-frame reference motion given input. For the motion imitation task, the input is defined as $\boldsymbol{o}_t^{\text{imitation}} \triangleq (\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}})$, consisting of the proprioception $\boldsymbol{s}_t^{\text{p}}$ and imitation goal $\boldsymbol{s}_t^{\text{g-mimic}}$ (one frame difference between current pose and reference motion. We follow PHC's [169] state, reward, and policy design to train one policy to imitate all motion sequences from the AMASS dataset. To increase the dexterity of the policy, we follow Omnigrasp [167] and augment the AMASS dataset's whole body motion with randomly selected hand motion from the Interhand [189] and GRAB [266] dataset. To transition from Isaac-

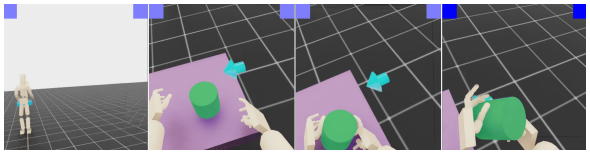


(a) Left hand: Object transport. 3^{rd} person and agent's camera view.



(b) Right hand: Transport and release.





(c) Both hands: Picking up large objects. Agent's camera view.

Figure 7.7. **Tabletop:** PDC is instructed directly from visual signals. A visual (top left and/or right) indicates in purple whether the corresponding hand should be prepared for contact. Changing to dark-blue indicates contact should be made. Instructing the agent to use both hands enables it to transport larger objects. Changing the indicator back to purple instructs the agent to release the object.

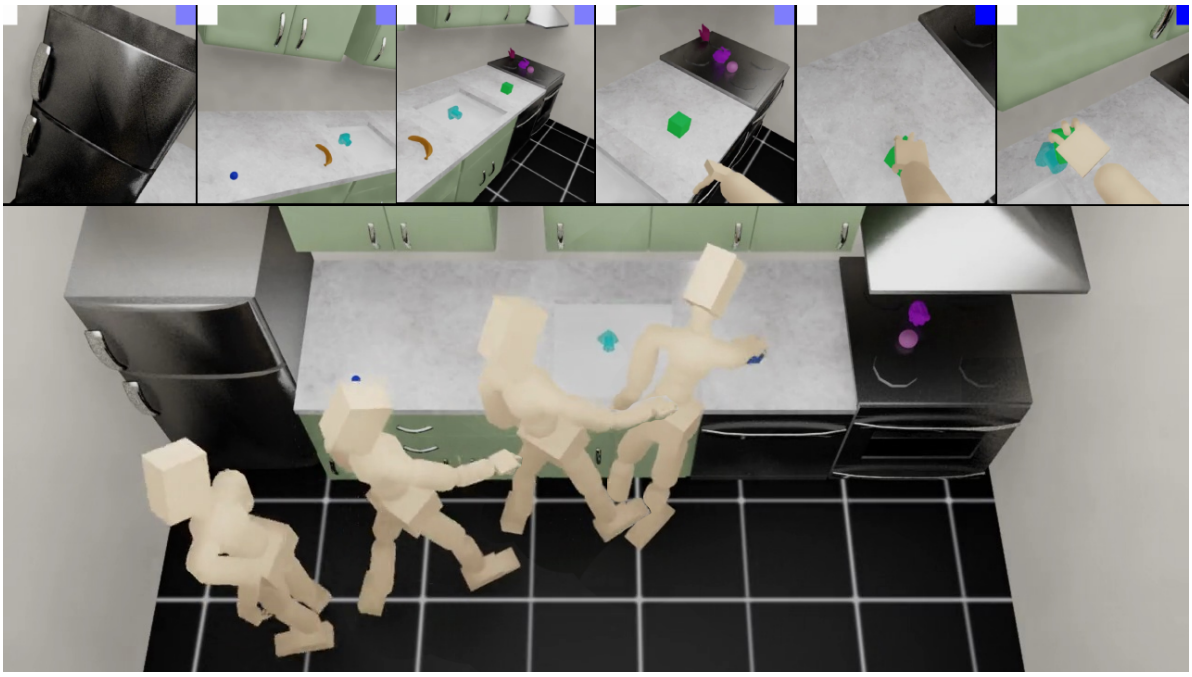
Gym [177] to IsaacSim (the underlying simulation engine for IsaacLab [185]), we implement the PHC-X policy in IsaacLab and train it for 5 days across using 8 GPUs. We report success rate, mean per joint position error $E_{\text{g-mpjpe}}$ (mm), local joint position error E_{mpjpe} (mm), acceleration error E_{acc} (mm/frame^2), and velocity error E_{vel} (mm/frame) to evaluate the performance of our policies. All metrics are completed between the simulated humanoid motion and the reference motion. From Table 7.6, we can see the our IsaacLab implementation achieves comparable motion imitation results as the ones in IsaacGym.

Humanoid Motion Prior, PULSE-X. PHC-X learns the motor skills required to perform most of the actions in the AMASS dataset, and then PULSE-X learns a CVAE-like motion latent space. Specifically, PULSE learns the encoder $\mathcal{E}_{\text{PULSE-X}}$, decoder $\mathcal{D}_{\text{PULSE-X}}$, and prior $\mathcal{P}_{\text{PULSE-X}}$ to compress motor skills into a latent representation. The encoder $\mathcal{E}_{\text{PULSE-X}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g-mimic}}) \sim \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^e, \boldsymbol{\sigma}_t^e)$ computes the latent code distribution based on current input states. The decoder $\mathcal{D}_{\text{PULSE-X}}(\boldsymbol{a}_t|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{z}_t)$ produces action (joint actuation) based on the latent code $\boldsymbol{z}_t$. The prior $\mathcal{P}_{\text{PULSE-X}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}}) \sim \mathcal{N}(\boldsymbol{z}_t|\boldsymbol{\mu}_t^p, \boldsymbol{\sigma}_t^p)$ defines a Gaussian distribution based on proprioception and replaces the unit Gaussian distribution used in VAEs [129]. The prior increases the expressiveness of the latent space and guides downstream task learning by forming a residual action space.

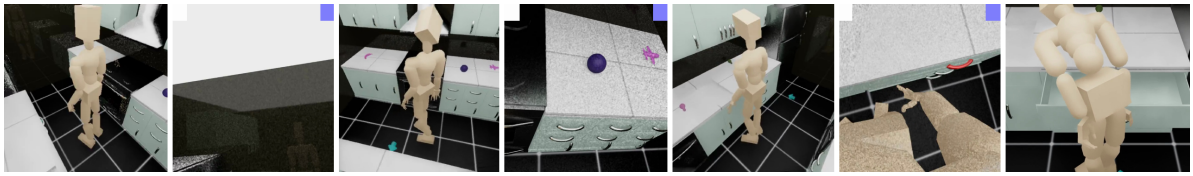
After the encoder, decoder, and prior is trained, similar to downstream task policies in PULSE, we form the action space of $\pi_{\text{Omnigrasp}}$ as the residual action with respect to prior's mean $\boldsymbol{\mu}_t^p$ and compute the PD target $\boldsymbol{a}_t$:

$$\boldsymbol{a}_t = \mathcal{D}_{\text{PULSE-X}}(\pi_{\text{Omnigrasp}}(\boldsymbol{z}_t^{\text{omnigrasp}}|\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}}) + \boldsymbol{\mu}_t^p, \boldsymbol{s}_t^{\text{p}}), \tag{7.3}$$

where $\boldsymbol{\mu}_t^p$ is computed by the prior $\mathcal{P}_{\text{PULSE-X}}(\boldsymbol{z}_t|\boldsymbol{s}_t^{\text{p}})$. The policy $\pi_{\text{Omnigrasp}}$ computes $\boldsymbol{z}_t^{\text{omnigrasp}} \in \mathcal{R}^{48}$ instead of the target $\boldsymbol{a}_t \in \mathcal{R}^{51\times 3}$ directly, and leverages the latent motion representation of PULSE-X to produce human-like motion. Table 7.6 shows that our implementation achieves a high success rate on the AMASS dataset.



(a) Object transport.



(b) Drawer opening.

Figure 7.8. **Emergent search in the kitchen task:** By training the agent in diverse and complex scenes, it learns generalizable behaviors and avoids overfitting. We observe that behaviors such as searching emerge. When the object is not in view, the agent scans the counter top and top drawers in order to learn of its objective and execute on it.

Method	# Envs	Learning Rate	σ	γ	ϵ	# of samples
PHC-X	3072×8	2×10^{-5}	0.05	0.99	0.2	$\sim 10^{10}$
PULSE-X	3072×8	5×10^{-4}	–	–	–	$\sim 10^{9}$
Omnigrasp	2048×8	2×10^{-5}	0.36	0.99	0.2	$\sim 10^{9}$
PDC	192×8	2×10^{-5}	0.36	0.99	0.2	$\sim 10^{9}$

Table 7.7. Hyperparameters for PDC, PHC-X, and PULSE-X. σ: fixed variance for policy. γ: discount factor. ϵ: clip range for PPO.

7.6.9 Details about PDC

Hyperparameters. Hyperparameters for training PHC-X, PULSE-X, and PDC can be found in Table 7.7. We do not change the hyperparameters significantly between training the visual policy and state-space policy, demonstrating the robustness of PPO across different tasks.

Rewards. For our reward,

$$r_t^{\text{PDC}} = \begin{cases} r_t^{\text{approach}} + r_t^{\text{lookat}}, & \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 > 0.2 \text{ and } t < \lambda_{\text{start}} \\ r_t^{\text{pre-grasp}} + r_t^{\text{lookat}}, & \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 \leq 0.2 \text{ and } t < \lambda_{\text{start}} \\ r_t^{\text{obj}} + r_t^{\text{lookat}}, & \lambda_{\text{start}} \leq t < \lambda_{\text{end}} \\ (1 - \mathbf{1}_{\text{has-contact}}), & t \geq \lambda_{\text{end}} \end{cases} \tag{7.4}$$

λ_{start} indicates the frame in which grasping should occur, and λ_{end} is when the frame should end and the object released. $\boldsymbol{p}_t^{\text{hand}}$ indicates the hands' position. When the object is far away from the hands ($\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 > 0.2$) and before grasping should start, the approach reward r_t^{approach} is similar to a point-goal [169, 299] reward $r_t^{\text{approach}} = \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 - \|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_{t-1}^{\text{hand}}\|_2$,, where the policy is encouraged to get close to the pre-grasp. After the hands are close enough ($\leq$ 0.2m), we use a more precise hand imitation reward: $r_t^{\text{pre-grasp}} = w_{\text{hp}} e^{-100\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \boldsymbol{p}_t^{\text{hand}}\|_2 \times \mathbb{1}\{\|\hat{\boldsymbol{p}}^{\text{pre-grasp}} - \hat{\boldsymbol{p}}_t^{\text{obj}}\|_2 \leq 0.2\}} + w_{\text{hr}} e^{-100\|\hat{\boldsymbol{\theta}}^{\text{pre-grasp}} - \boldsymbol{\theta}_t^{\text{hand}}\|_2}$, to encourage the hands to be close to pre-grasps. After the grasp start time, we switch to the object 3D location reward $r_t^{\text{obj}} = e^{-5\|\boldsymbol{p}_t^{\text{target}} - \boldsymbol{p}_t^{\text{obj}}\|_2} \times \mathbf{1}_{\text{correct-hand}}$ to encourage the object being moved to specific locations. $\mathbf{1}_{\text{correct-hand}}$ is the indicator random variable to determine if the correct hand has contact with the object. After the grasping should end ($t \geq \lambda_{\text{end}}$), we encourage the agent to not be in contact with the object: $\mathbf{1}_{\text{has-contact}}$ determines if any hand has contact with the object. The drawer open reward r_t^{drawer} is defined as how many degrees (angles defined by IsaacLab) the drawer is opened, clipped to 0 and 1.

Network Architecture . For our networks, we uses a two-layer CNN with each with 32 channels. Our CNN produces a latent feature space of $\mathcal{R}^{128}$. We use 6-layer MLPs of units (2048, 2048, 1024, 1024, 512, 512) with silu activation [69]. Our GRU has one layer and 128 hidden units.

7.6.10 Details about Dataset

We use the same train/test sequence split from Omnigrasp [167] and Braun *et al*. [26]. For training, we use a subset of the objects and pick the ones that are more common in the households. Specifically, we use the following 25 categories:

`torusmedium`	`flashlight`	`bowl`
`lightbulb`	`alarmclock`	`hammer`
`scissors`	`cylindermedium`	`stapler`
`phone`	`duck`	`airplane`
`knife`	`cup`	`wineglass`
`fryingpan`	`cylindersmall`	`waterbottle`
`banana`	`mouse`	`flute`
`headphones`	`stamp`	`toothbrush`
`watch`		

Out of the 1016 training sequences from GRAB, 571 is picked. For testing, we use the same 140 sequences and object types are `apple`, `binoculars`, `camera`, `mug`, `toothpaste`.

7.6.11 Supplementary Results

7.6.12 Per-object Success Rate

In Table 7.4 and Table 7.5, we report the per-object success rate for our stereo policy in the tabletop and kitchen scene on the grab dataset. From the result, we can see that objects such as bowl and waterbottle

are easy to grasp, while objects with irregular shapes like airplanes are harder. Small and thin objects like knife is also hard to pick up and has one of the lowest success rate across the kitchen and tabletop scene. Test objects such as binoculars are harder since they are bigger and difficult to grasp with one hand. The kitchen grasping paints a similar picture. No object has a zero success rate.

7.6.13 Additional Qualitative Results

In Figure 7.7 and Figure 7.8 we present additional qualitative examples of the resulting controllers. They showcase the ability of the agent to handle objects in diverse scenes, using on-screen indicators as instructions. By training directly from vision within complex and diverse tasks, we observe that search and dexterity emerge.

7.7 Discussion

Failure Cases and Limitations. The vision-centric design in PDC leads to emergent search behaviors and success rates that outperform the state-based benchmark. However, the kitchen scene provides some challenging scenarios. For example, as the agent reaches towards objects located near the wall (on the far end of the counter), low-hanging cabinets and the steam-collector may interfere with the agent's vision. The lack of visual clarity may lead to failing to grasp the object. In addition, our agent is trained on a specific sequence of reach towards – then grasp. If the grasp fails, it does not attempt to re-grasp. Instead, it shifts its focus to tracking the marker, despite not holding the object. Our policy also tends to shake its head from time to time and could benefit from image-stabilization techniques. Future work may overcome these limitations by providing additional and looser rewards. Such as enabling re-grasping, or using more advanced vision analysis to ensure the object is in view instead of naive angle comparison.

Conclusions. We propose PDC, a framework for learning multiple vision-guided tasks using a single policy via perception-as-interface and visual reinforcement learning. We observe that training in complex and diverse scenes leads to the emergence of behaviors such as search. As one of the first to tackle visual dexterous humanoid control, we believe our method and task setting open many doors for future research. Our perception-as-interface and proprioception-only policy also has the potential to be adopted to real humanoids.

Part IV

Transferring to Real Humanoids

Chapter 8

H2O: Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation

8.1 Introduction

In this chapter, We aim transfer PHC's motion imitation pipeline to a full-sized humanoid robot. This can enable real-time teleoperation of a full-sized humanoid robot by a human teleoperator using an RGB camera. Humanoid robots, with their physical form closely mirroring that of humans, present an unparalleled opportunity for real-time teleoperation. This alignment of the embodiment allows for a seamless integration of human cognitive skills with versatile humanoid capabilities [59]. Such synergy stimulated by human-to-humanoid teleoperation is crucial for complex tasks (*e.g.*, household chores, medical assistance, high-risk rescue operations) that are yet too challenging for a fully autonomous robot, but possible for existing hardware teleoperated by humans [48, 77]. In this work, we transfer human motions to humanoid behaviors in a real-time fashion using an RGB camera. This system also has the potential to enable large-scale and high-quality data collection of human operations for robotics [77,343], where human-teleoperated actions can be used for imitation learning.

However, whole-body control of full-sized humanoids is a long-standing problem in robotics [136], and complexity increases when controlling the humanoid to replicate free-form human movements in real-time [59]. Existing work on whole-body humanoid teleoperation has achieved remarkable results via model-based controllers [115, 116, 186, 226], but they all use simplified models due to the high computational cost of modeling the full dynamics of the system [314, 337], which limits the scalability to dynamic motions. Furthermore, these works are highly dependent on contact measurement [64, 198], leading to reliance on external setups such as the exoskeleton [116] and force sensors [226, 227] for teleoperation.

Recent advances in reinforcement learning (RL) for humanoid control provide a promising alternative. First, in the graphics community, RL has been used to generate complex human movements [208, 297], perform a variety of tasks [213], and track real-time human motions captured by a webcam [169] in simulation. However, due to unrealistic state-space design and partial disregard of the hardware limit (*e.g.* torque / joint limit), it remains a question whether these methods can be applied to a full-sized humanoid. On the other hand, RL has achieved robust and agile biped locomotion in the real world [148, 223, 253]. To date, however, there has been no existing work on RL-based whole-body humanoid teleoperation. The most closely related effort is a concurrent study [43], which focuses on learning to replicate upper-body motions and uses root velocity tracking for the lower body, from offline human motions rather than real-time teleoperation.

In this chapter, we design a complete system for humanoid teleportation in real time. First, we identify one of the primary challenges in whole-body humanoid teleoperation as the lack of a dataset with feasible motions tailored to the humanoid, which is essential for training a controller that can track diverse motions. Although direct human-to-humanoid retargeting has been explored in previous locomotion-focused efforts [52, 60, 224], retargeting a large-scale human motion dataset to the humanoid presents new challenges. That is, the significant dynamics discrepancy between humans and humanoids means that some human motions could be infeasible for the humanoid (*e.g.* cartwheeling, steps wider than the leg lengths of the humanoid). In light of this, we introduce an automated "sim-to-data" process to retarget and refine a large-scale human motion dataset [176] into motions that are feasible for real-world humanoid embodiment. Specifically, we first retarget the human motions to the humanoid via inverse kinematics, and train a humanoid controller with access to privileged state information [169] (PHC) to imitate the unfiltered motions in simulation. Afterwards, we remove the motion sequences that the privileged imitator fails to track. By doing so, we create a large-scale humanoid-compatible motion dataset.

After obtaining a dataset of feasible motions, we develop a scalable training process for the real-world motion imitator that incorporates extensive domain randomization to bridge the sim-to-real gap. To facilitate real-time teleoperation, we design a state space that prioritizes the inputs available in the real world using an RGB camera, such as the keypoint positions. During inference, we use an off-the-shelf human pose estimator [143] to provide global human body positions for the humanoid to track.

In summary, we demonstrate the feasibility of an RL-based real-time Human-to-Humanoid (H2O) teleoperation system. Our contributions include:

- A scalable retargeting and "sim-to-data" process to obtain a large-scale motion dataset feasible for the real-world humanoid robot;
- Sim-to-real transfer of the RL-based whole-body tracking controller that scales to a large number of motions;
- A real-time teleoperation system with an RGB camera and 3D human pose estimation, demonstrating fulfillment of various whole-body motions including walking, pick-and-place, stroller pushing, boxing, hand-waving, ball kicking, etc.

8.2 Related Works

Physics-Based Animation of Human Motions. Physics-based simulation has been used to generate realistic and natural motions for avatars [79, 169, 170, 208, 211, 213, 215, 292, 295, 297]. With motion capture as the main source of human motion data [176], RL is often used to learn avatar controllers that can mimic these motions, offering distinctive styles [208, 215], scalability [169, 172, 297], and reusability [170, 213].

However, realistic animation in physics-based simulators does not guarantee real-world applicability, especially for humanoids. Simulated humanoid avatars typically have high degrees of freedom and large joint torques [30], and sometimes need non-physical assistive external forces [327]. In this work, we demonstrate that, with carefully designed sim-to-real training, approaches in the humanoid animation community can be applied to a real-world humanoid robot.

Transferring Human Motions to Real-World Humanoids. Before the emergence of RL-based humanoid controllers, traditional methods typically employ model-based optimization to track retargeted motions while maintaining stability [59]. To this end, these methods minimize tracking errors under the constraints of stability and contacts, requiring predefined contact states [64, 134, 186, 198, 206, 207, 228, 314] or estimated contacts from sensors [14, 107, 115, 116, 227], hindering large-scale deployment outside the laboratory. [337] use contact-implicit model predictive control (MPC) to track motions extracted from videos, but trajectories must first be optimized offline to ensure dynamic feasibility. Furthermore, the

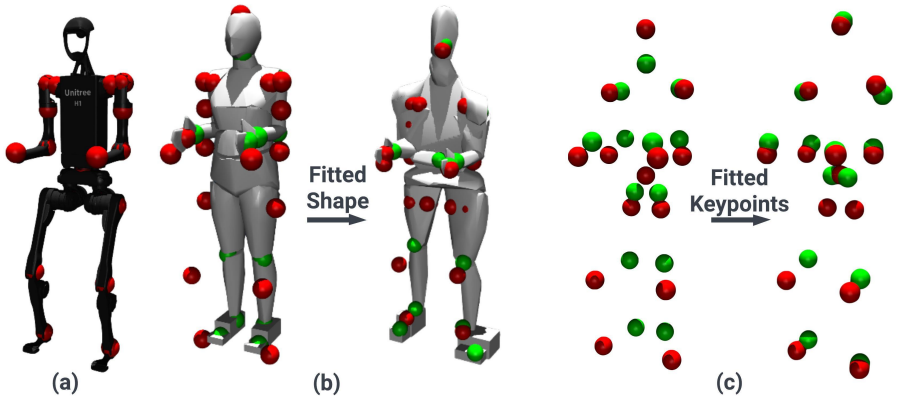


Figure 8.1. Fitting the SMPL body to the H1 humanoid. (a) Visualization of the humanoid keypoints (red dots) (b) Humanoid keypoints vs SMPL keypoints (green dots and mesh) before and after fitted SMPL shape β'. (c) Corresponding 12 joint position before and after fitting.

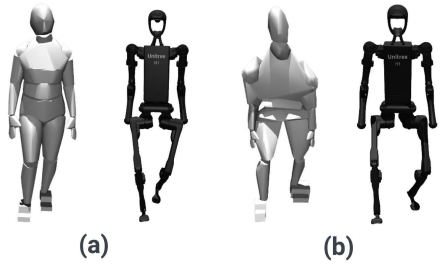


Figure 8.2. Effect of using a fitted SMPL shape β' instead of mean body shape on position-based retargeting. (a) Retargting without using β', which results in unstable "in-toed" humanoid motion. (b) Retargeting using β', which result in balanced humanoid motion.

model used in MPC needs to be simplified due to computational burden [186,227,337], which limits the capabilities.

RL-based controllers may provide an alternative that does not require explicit contact information. Some works [24,269] use imitation learning to transfer human-style motions to the controller but do not accurately track human motions. [43] train whole-body humanoid controllers that can replicate upper body movements from offline human motions, but the lower body relies on velocity tracking and does not mimic human lower body movements. In comparison, our work achieves real-time whole-body tracking of human motions.

Teleoperation of Humanoids. Teleoperation of humanoids can be categorized into three types: 1) task-space teleoperation [55,246], 2) upper-body-retargeted teleoperation [36,70], and 3) whole-body teleoperation [107,115,186,198,264]. For the first and second types, the shared morphology between humans and humanoids is not fully utilized, and whole-body control must be solved in a task-specified way. This also raises the concern that, if tracking lower body movement is not necessary, the robot could opt for designs with better stability, such as a quadruped [16] or wheels [139].

Our work belongs to the third type and is the first to achieve learning-based whole-body teleoperation. Moreover, our approach does not require capture markers or force sensors on the human teleoperator, as we directly employ an RGB camera to capture human motions for tracking, potentially paving the way for collecting large-scale humanoid data for training autonomous agents.

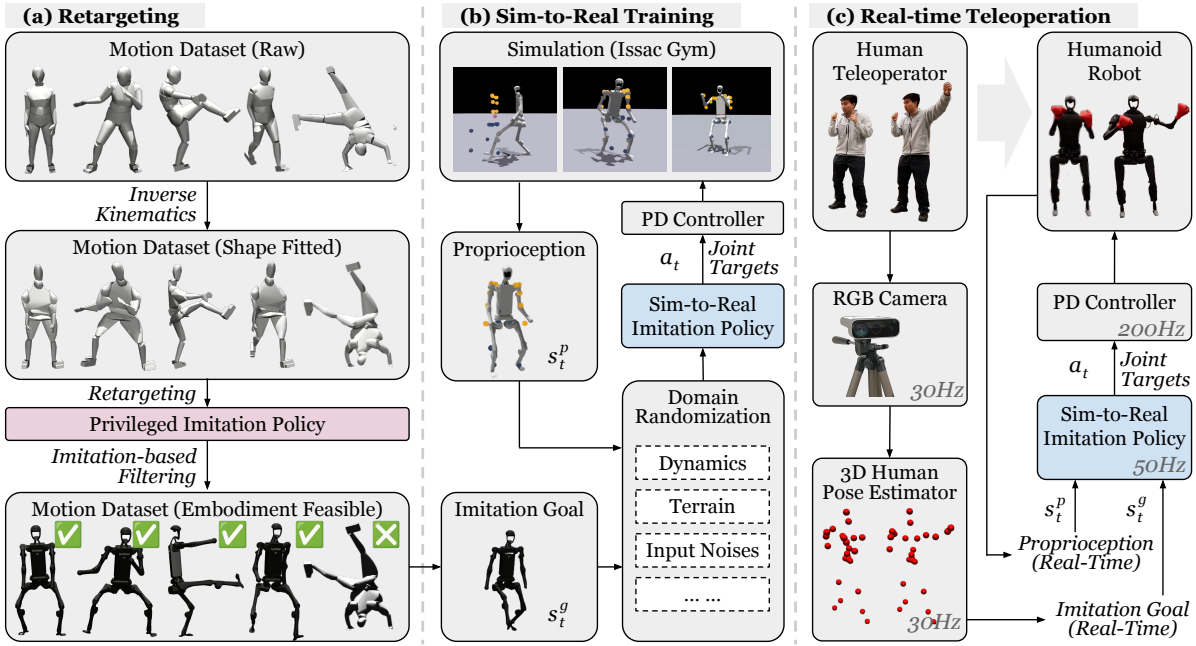


Figure 8.3. Overview of H2O: (a) **Retargeting** (Section 8.3): H2O first aligns the SMPL body model to a humanoid's structure by optimizing shape parameters. Then H2O retargets and removes the infeasible motions using a trained privileged imitation policy, producing a clean motion dataset. (b) **Sim-to-Real Training**: (Section 8.4) An imitation policy is trained to track motion goals sampled from a cleaned dataset. (c) **Real-time Teleoperation Deployment** (Section 8.5.2): The real-time teleoperation deployment captures human motion through an RGB camera and a pose estimator, which is then mimicked by a humanoid robot using the trained sim-to-real imitation policy.

8.3 Retargeting Human Motions for Humanoid

To enable humanoid motion imitation for unscripted human motion, we require a large amount of whole-body motion to train a robust motion imitation policy. Since humans and humanoids also have a nontrivial difference in body structure, shape, and dynamics, naively retargeted motion from a human motion dataset can result in a large number of motions impossible for our humanoid to perform. These imfeasible motion sequences can hinder imitation training as observed in prior work [170]. To resolve these issues, we design a "sim-to-data" approach to complement traditional retargeting to convert a large-scale human motion dataset to feasible motions for humanoids.

8.3.1 Motion Retargeting

As there is a non-trivial difference between the SMPL kinematic structure and the humanoid kinematic tree, we perform a two-step process for the initial retargeting. First, since the SMPL body model can represent different body proportions, we first find a body shape β' closest to the humanoid structure. We choose 12 joints that have a correspondence between humans and humanoids, as shown in Fig.8.1 and perform gradient descents on the shape parameter $\boldsymbol{s}$ to minimize the joint distances using a common rest pose. After finding the optimal β', given a sequence of motions expressed in SMPL parameters, we use the original translation $\boldsymbol{p}$ and pose $\boldsymbol{\theta}$, but the fitted shape β' to obtain the set of body keypoint positions. Then we retarget motion from human to humanoid by minimizing the 12 joint position differences using Adam optimizer [128]. Notice that our retargeting process try to match the end effectors of the human to the humanoid (*e.g.* ankles, elbows, wrists) to preserve the overall motion pattern. Another approach

is direct copying the local joint angles from human to humanoid, but that approach can lead to large differences in end-effector positions due to the large difference in kinematic trees. During this process, we also add some heuristic-based filtering to remove unsafe sequences, such as sitting on the ground. The motivation to find β' before retargeting is that in the rest pose, our humanoid has a large gap between its feet. If naively trying to match the foot movement between the human and the humanoid, the humanoid motion can have an in-toed artifact. Using β', we can find a human body structure has a large gap between its rest pose (as shown in Fig.8.1). Using β' during fitting can effectively create motion that is more feasible for the humanoid, as shown in Fig.8.2. From the AMASS dataset $\hat{\boldsymbol{Q}}$ that contains 13k motion sequences, this process computes 10k retargted motion sequences $\hat{\boldsymbol{Q}}^{\text{retarget}}$.

8.3.2 Simulation-based Data Cleaning

As shown in Figure 8.3, $\hat{\boldsymbol{Q}}^{\text{retarget}}$ contains a large number of implausible motions for the humanoid due to the significant gap between the capabilities of a human and a motor-actuated humanoid. Manually finding these data sequences from a large-scale dataset can be a rather cumbersome process. Thus, we propose a "sim-to-data" procedure, where we train a motion imitator $\pi_{\text{privileged}}$ (similar to PHC [169]) with access to privileged information and no domain randomization to imitate all uncleaned data $\hat{\boldsymbol{Q}}^{\text{retarget}}$. Without domain randomization, $\pi_{\text{privileged}}$ can perform well in motion imitation, but is not suitable for transfer to the real humanoid. However, $\pi_{\text{privileged}}$ represents the upper bound of motion imitation performance, and sequences which $\pi_{\text{privileged}}$ fails to imitate represent implausible ones. Specifically, we train $\pi_{\text{privileged}}$ following the same state space, control parameters, and hard-negative mining procedure proposed in PULSE [170], and train a single imitation policy to imitate the entire retargeted dataset. After training, ~8.5k out of 10k motion sequences from AMASS turn out to be plausible for the H1 humanoid, and we denote the obtained clean dataset as $\hat{\boldsymbol{Q}}^{\text{clean}}$.

Privileged Motion Imitation Policy. To train $\pi_{\text{privileged}}$, we follow PULSE [170] and train a motion imitator with access to the full rigid body state of the humanoid. Specifically, for the privileged policy $\pi_{\text{privileged}}$, its proprioception is defined as $\boldsymbol{s}_t^{\text{p-privileged}} \triangleq [\boldsymbol{p}_t, \boldsymbol{\theta}_t, \boldsymbol{v}_t, \boldsymbol{\omega}_t]$, which contains the global 3D rigid body position $\boldsymbol{p}_t$, orientation $\boldsymbol{\theta}_t$, linear velocity $\boldsymbol{v}_t$, and angular velocity $\boldsymbol{\omega}_t$ of all rigid bodies in the humanoid. The goal state is defined as $\boldsymbol{s}_t^{\text{g-privileged}} \triangleq [\hat{\boldsymbol{\theta}}_{t+1} \ominus \boldsymbol{\theta}_t, \hat{\boldsymbol{p}}_{t+1} - \boldsymbol{p}_t, \hat{\boldsymbol{v}}_{t+1} - \boldsymbol{v}_t, \hat{\boldsymbol{\omega}}_{t+1} - \boldsymbol{\omega}_t, \hat{\boldsymbol{\theta}}_{t+1}, \hat{\boldsymbol{p}}_{t+1}]$, which contains the one-frame difference between the reference and current simulation result for all rigid bodies on the humanoid. It also contains the next frame's reference rigid body orientation and position. All values are normalized to the humanoid's coordinate system. Notice that all values are global, and values such as global rigidbody linear velocity $\boldsymbol{v}_t$ and angular velocity $\boldsymbol{\omega}_t$ are hard to obtain accurately in the real world.

8.4 Whole-Body Teleoperation Policy Training

8.4.1 State Space

To achieve real-time teleoperation of humanoid robots, the state space of RL policy must contain only quantities available in the real world. This differs from the simulation-only approaches, where all the physics information (*e.g.*, foot contact force) is available. For example, in the real world, we have no access to each joint's precise global angular velocity due to the lack of IMUs, but the privileged policy $\pi_{\text{privileged}}$ requires them.

In our state space design, proprioception is defined by $\boldsymbol{s}_t^{\text{p}} \triangleq [\boldsymbol{q}_t, \dot{\boldsymbol{q}}_t, \boldsymbol{v}_t^{root}, \boldsymbol{\omega}_t^{root}, \boldsymbol{g}_t, \boldsymbol{a}_{t-1}]$, with joint position $\boldsymbol{q}_t \in \mathbb{R}^{19}$ (DoF position), joint velocity $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{19}$ (DoF velocity), root linear velocity $\boldsymbol{v}_t^{root} \in \mathbb{R}^3$, root angular velocity $\boldsymbol{\omega}_t^{root} \in \mathbb{R}^3$, root projected gravity $\boldsymbol{g}_t \in \mathbb{R}^3$, and last action $\boldsymbol{a}_{t-1} \in \mathbb{R}^{19}$. The goal

Term	Expression	Weight
	Penalty	
Torque limits	$\mathbb{1}(\boldsymbol{\tau}_t \notin [\boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max}])$	$-2e^{-1}$
DoF position limits	$\mathbb{1}(\boldsymbol{q}_t \notin [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}])$	$-1e^{2}$
Termination	$\mathbb{1}_{\text{termination}}$	$-2e^{2}$
	Regularization	
DoF acceleration	$\|\ddot{\boldsymbol{q}}_t\|_2^2$	$-8.4e^{-6}$
DoF velocity	$\|\dot{\boldsymbol{q}}_t\|_2^2$	$-3e^{-3}$
Action rate	$\|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|_2^2$	$-9e^{-1}$
Torque	$\|\boldsymbol{\tau}_t\|$	$-9e^{-5}$
Feet air time	$T_{\text{air}} - 0.25$ [240]	$8e^{2}$
Feet contact force	$\|F_{\text{feet}}\|_2^2$	$-1e^{-1}$
Stumble	$\mathbb{1}(F_{\text{feet}}^{xy} > 5 \times F_{\text{feet}}^{z})$	$-1e^{3}$
Slippage	$\|\boldsymbol{v}_t^{\text{feet}}\|_2^2 \times \mathbb{1}(F_{\text{feet}} \geq 1)$	$-3e^{1}$
	Task Reward	
DoF position	$\exp(-0.25\|\hat{\boldsymbol{q}}_t - \boldsymbol{q}_t\|_2)$	$2.4e^{1}$
DoF velocity	$\exp(-0.25\|\hat{\dot{\boldsymbol{q}}}_t - \dot{\boldsymbol{q}}_t\|_2^2)$	$2.4e^{1}$
Body position	$\exp(-0.5\|\boldsymbol{p}_t - \hat{\boldsymbol{p}}_t\|_2^2)$	$4e^{1}$
Body rotation	$\exp(-0.1\|\boldsymbol{\theta}_t \ominus \hat{\boldsymbol{\theta}}_t\|)$	$1.6e^{1}$
Body velocity	$\exp(-10.0\|\boldsymbol{v}_t - \hat{\boldsymbol{v}}_t\|_2)$	$6e^{1}$
Body angular velocity	$\exp(-0.01\|\boldsymbol{\omega}_t - \hat{\boldsymbol{\omega}}_t\|_2)$	$6e^{1}$

Table 8.1. Reward components and weights: penalty rewards for preventing undesired behaviors for sim-to-real transfer, regularization to refine motion, and task reward to achieve successful whole-body tracking in real-time.

state is $\boldsymbol{s}_t^{\text{g}} \triangleq [\hat{\boldsymbol{p}}_t^{\text{kp}}, \hat{\boldsymbol{p}}_t^{\text{kp}} - \boldsymbol{p}_t{}^{\text{kp}}, \hat{\dot{\boldsymbol{p}}}_t^{\text{kp}}]$. $\hat{\boldsymbol{p}}_t^{\text{kp}} \in \mathbb{R}^{8\times 3}$ is the position of eight selected reference body positions (shoulders, elbows, hands, ankles); $\hat{\boldsymbol{p}}_t^{\text{kp}} - \boldsymbol{p}_t{}^{\text{kp}}$ is the position difference between the reference joints and humanoid's own joints; $\hat{\dot{\boldsymbol{p}}}_t^{\text{kp}}$ is the linear velocity of the reference joints. All values are normalized to the humanoid's own coordinate system. As a comparison, we also consider a reduced goal state $\boldsymbol{s}_t^{\text{g-reduced}} \triangleq (\hat{\boldsymbol{p}}_t^{\text{kp}})$, where only the reference position $\hat{\boldsymbol{p}}_t^{\text{kp}}$ but not the position difference. The action space of the agile policy consists of 19-dim joint targets.

8.4.2 Reward Design

We formulate the reward function r_t with the summation of three terms: 1) penalty; 2) regularization; and 3) task rewards, which are summarized in detail in Table 8.1. Note that while we only have eight selected body positions $\hat{\boldsymbol{p}}_t^{\text{kp}}$ in our state space, we provide six *full-body* reward terms for all joints (DoF position, DoF velocity, body position, body rotation, body velocity, body angular velocity) for the imitation task. These expressive rewards give more dense reward signals for efficient RL training.

8.4.3 Domain Randomization

Domain randomization has been shown to be the key source of robustness and generalization to achieve successful sim-to-real transfers [148, 209]. All the domain randomization we use in H2O are listed in Table 8.2, including ground friction coefficient, link mass, Centor-of-Mass (CoM) position of the torso

Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.2, 1.1)$
Base CoM offset	$\mathcal{U}(-0.1, 0.1)$m
Link mass	$\mathcal{U}(0.7, 1.3) \times$ default kg
P Gain	$\mathcal{U}(0.75, 1.25) \times$ default
D Gain	$\mathcal{U}(0.75, 1.25) \times$ default
Torque RFI [32]	$0.1 \times$ torque limit N $\cdot$ m
Control delay	$\mathcal{U}(20, 60)$ms
External Perturbation	
Push robot	interval $= 5s$, $v_{xy} = 0.5$m/s
Randomized Terrain	
Terrain type	flat, rough, low obstacles [100]

Table 8.2. The range of dynamics randomization. Describing simulated dynamics randomization, external perturbation, and randomized terrain, which are important for sim-to-real transfer and boost robustness and generalizability.

link, PD gains of the PD controller, torque noise on the actually applied torques on each joint, control delay, terrain types. The link mass and PD gains are independently randomized for each link and joint, and the rest are episodic randomized. These domain randomization together can effectively facilitate the sim-to-real transfer for the real-world dynamics and hardware gaps.

8.4.4 Early Termination Conditions

We introduce three early termination conditions to make the RL training process more sample-efficient: 1) low height: the base height is lower than 0.3m; 2) orientation: the projected gravity on x or y axis exceeds 0.7; 3) teleoperation tolerance: the average link distance between the robot and reference motions is further than 0.5m.

8.5 Experimental Results

8.5.1 Simulation Experiments

Baselines. To reveal the effect of different retargeting, state space designs, and sim-to-real training techniques on whole-body teleoperation performance, we consider four baselines:

1. Privileged policy $\pi_{\text{privileged}}$: The privileged policy (trained without any sim-to-real regularizations or domain randomizations) is used to filter the dataset to find infeasible motion. It has no sim-to-real capability and has a much higher input dimension.
2. H2O-w/o-sim2data: H2O without the "sim-to-data" retargeting, trained on the $\hat{\boldsymbol{Q}}^{\text{retarget}}$;
3. H2O-reduced: H2O with a state space of goal state consisting only of selected body positions $\boldsymbol{s}_t^{\text{g-reduced}}$.
4. H2O: Our full H2O system, with all the retargeting process introduced in Section 8.3 and the state space design introduced in Section 8.4.1, trained on $\hat{\boldsymbol{Q}}^{\text{clean}}$;

Metrics. We evaluate these baselines in simulation on the uncleaned retargeted AMASS dataset (10k sequences $\hat{\boldsymbol{Q}}^{\text{retarget}}$). The metrics are as follows:

			All sequences					Successful sequences			
Method	State Dimension	Sim2Real	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
Privileged policy	$\mathcal{S} \subset \mathcal{R}^{778}$	✗	85.5%	50.0	43.6	6.9	7.8	46.0	40.9	5.2	6.2
H2O-reduced	$\mathcal{S} \subset \mathcal{R}^{90}$	✓	53.2%	200.2	115.8	11.2	13.8	182.5	111.0	3.0	8.1
H2O-w/o-sim2data	$\mathcal{S} \subset \mathcal{R}^{138}$	✓	67.9%	176.6	95.0	10.2	12.2	163.1	93.8	3.0	7.5
H2O	$\mathcal{S} \subset \mathcal{R}^{138}$	✓	**72.5%**	**166.7**	**91.7**	**8.9**	**11.0**	**151.0**	**88.8**	**2.9**	**7.0**

Table 8.3. Quantitative motion imtiation results the uncleaned retargeted AMASS dataset $\hat{\boldsymbol{Q}}^{\text{retarget}}$.

1. Success rate: the success rate (Succ) as in PHC [169], deeming imitation unsuccessful when, at any point during imitation, the average difference in body distance is on average further than 0.5m. Succ measures whether the humanoid can track the reference motion without losing balance or significantly lag behind.
2. E_{mpjpe} and $E_{g-\text{mpjpe}}$: the global MPJPE $E_{g-\text{mpjpe}}$ and the root-relative mean per-joint position error (MPJPE) E_{mpjpe} (in mm), measuring our imitator's ability to imitate the reference motion both globally and locally (root-relative).
3. E_{acc} and E_{vel}: To show physical realism, we also compare acceleration E_{acc} (mm/frame^2 and velocity E_{vel} (mm/frame) difference.

Results. The experimental results are summarized in Table 8.3, where H2O significantly outperforms H2O-w/o-sim2data and H2O-reduced by a large margin, demonstrating the importance of the "sim-to-data" process and the state-space design of motion goals for RL. Note that the privileged policy and H2O-w/o-sim2data are trained on the entire retargeted AMASS dataset $\hat{\boldsymbol{Q}}^{\text{retarget}}$ while H2O and H2O-reduced are trained on the filtered dataset $\hat{\boldsymbol{Q}}^{\text{clean}}$. The success rate gap between H2O and the privileged policy comes from two factors: 1) H2O uses a much more practical and less informative observation space compared to the privileged policy; 2) H2O is trained with all sim-to-real regularizations and domain randomization. These two factors will both lead to degradation in simulation performance. This shows that while the RL-based avatar control frameworks have achieved impressive results in simulation, transferring them to the real world requires more robustness and stability. With the carefully chosen dataset and the state space, we could make H2O achieve a higher success rate compared to H2O-w/o-sim2data and H2O-reduced. By comparing H2O with H2O-w/o-sim2data, we can see that our "sim-to-data" process is effective in obtaining higher success rate, even when the RL policy is trained on *less* data. Intuitively, an implausible motion may cause the policy to waste resources trying to achieve them, and filtering them out can lead to better overall performance, as also observed in PULSE [170]. Comparing H2O with H2O-reduced, the only difference is the design of the state space of the goal, which indicates that including more informative physical information about motions helps RL to generalize to large-scale motion imitation.

Ablation on Motion Dataset Size. To show how motion tracking performance scales with the size of the motion dataset, we test H2O with different size of $\hat{\boldsymbol{Q}}^{\text{clean}}$ by randomly selecting 0.1%, 1%, 10% of $\hat{\boldsymbol{Q}}^{\text{clean}}$. The results are summarized in Table 8.4, where policies trained larger motion datasets continue to improve the tracking performance. Notice that a policy trained on only 0.1% of the data can achieve a surprisingly high success rate, most likely due to the ample domain randomization applied to the humanoid, such as push robot significantly widens the state the humanoid has encountered, improving its generalization capability.

8.5.2 Real-world Demonstrations

Deployment Details. For real-world deployment tests, we use a standard 1080P webcam as the RGB

	All sequences				
Dataset Size	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel}
0.1% of $\hat{\boldsymbol{Q}}^{\text{clean}}$	52.0%	198.0	107.9	12.4	13.7
1% of $\hat{\boldsymbol{Q}}^{\text{clean}}$	58.8%	183.8	96.4	10.7	12.0
10% of $\hat{\boldsymbol{Q}}^{\text{clean}}$	61.3%	174.3	92.3	10.8	12.1
100% of $\hat{\boldsymbol{Q}}^{\text{clean}}$	**72.5%**	**166.7**	**91.7**	**8.9**	**11.0**

Table 8.4. Quantitative results of H2O on different sizes of motion dataset for training, evaluated on the uncleaned retargeted AMASS dataset $\hat{\boldsymbol{Q}}^{\text{retarget}}$.

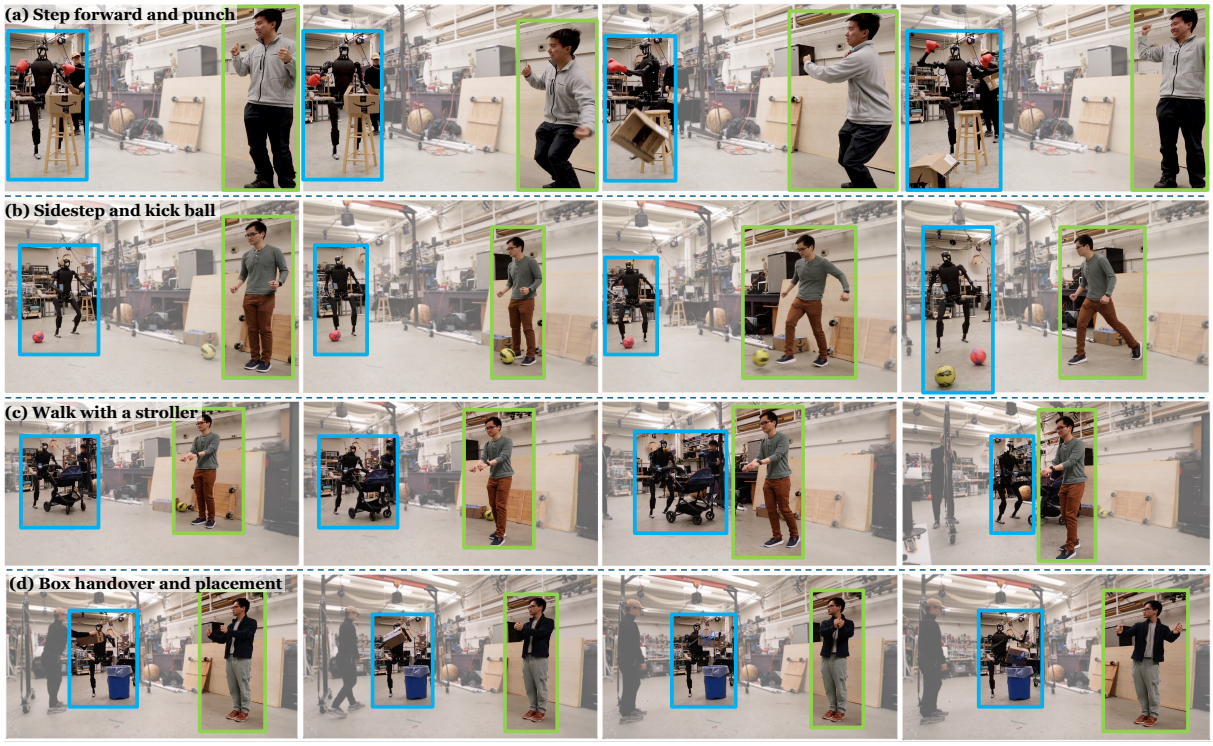


Figure 8.4. The humanoid robot is teleoperated in real-time using an RGB camera by the human teleoperator. (a) The humanoid mimics the human teleoperator, advancing one step while delivering a punch to displace a box, followed by a victory gesture. (b) The humanoid executes a precise sidestep to align with a ball and delivers a controlled kick using its right foot. (c) The humanoid demonstrates forward walking while pushing a stroller. (d) The operator teleoperates the humanoid to catch a box, rotate its waist, and drop the box into a waste bin. **Videos:** see the website.

camera, and use HybrIK [143] as the 3D human pose estimator running at 30Hz. For the linear velocity estimation of the robot, we leverage the motion capture system (50Hz), and all the other proprioception is obtained from built-in sensors (200Hz) of Unitree H1 humanoid. Linear velocity state estimation could be replaced by onboard visual/LiDAR odometry methods, though we opt in to MoCap for this work due to its simplicity.

Real-world Teleoperation Results. For real-time teleoperation, the 3D pose estimation from the RGB camera is noisy and can suffer from perspective bias, but our H2O policy shows a strong generalization

ability to real-world estimated motion goals in real-time. The real-world teleoperation is shown in Figure 8.4, Figure 8.5 and Figure 8.6, where H2O enables precise real-time teleoperation of humanoids to do whole-body dynamic motions like ball kicking, walking, and back jumping. More demonstrations can be found on our website.

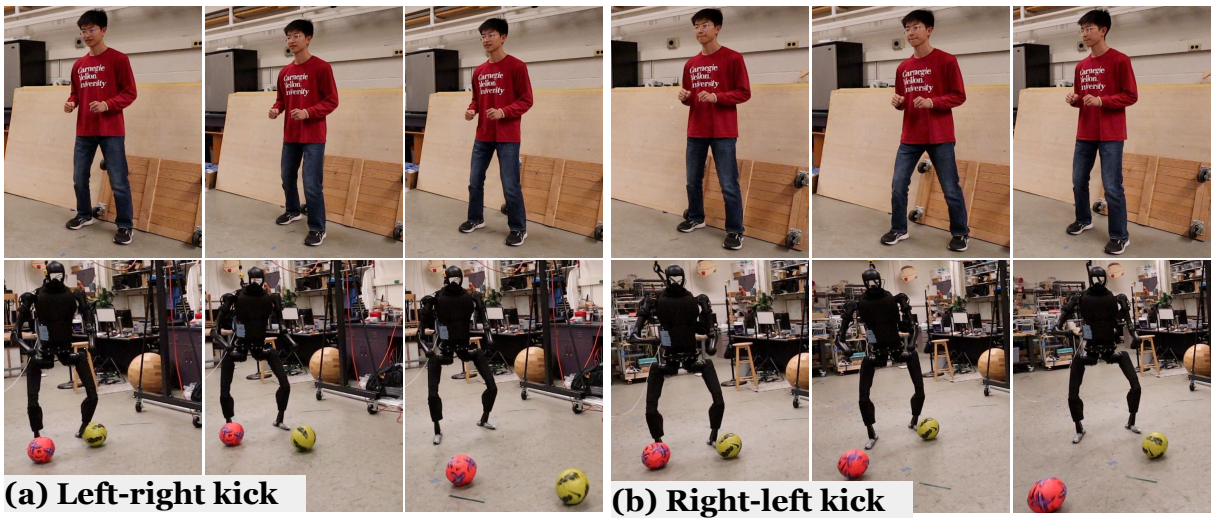


Figure 8.5. The humanoid robot is able to track the precise lower-body movements of the human teleoperator.

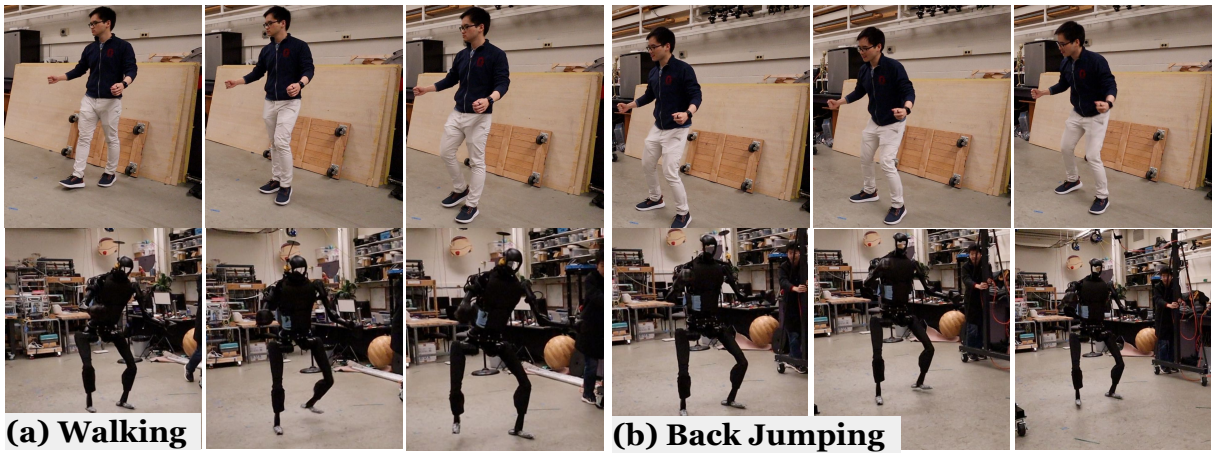


Figure 8.6. The humanoid robot is able to track walking motions of human-style pace and imitate continuous back jumping.

Robustness. Our H2O system can keep balance under external force disturbances, as shown in Figure 8.7. These tests demonstrate the robustness of our system.

8.6 Discussions, Limitations, and Future Work

Towards Universal Humanoid Teleoperation. Our ultimate goal is to enable the humanoid to follow as many human-demonstrated motions as possible. We emphasize three key factors that can be improved

Figure 8.7. Robustness Tests of our H2O system under powerful kicking. The policy is able to maintain balance for both stable and dynamic teleoperated motions.

in the future. 1) Closing the representation gap: as shown in Section 8.5.1, the state representation of the motion goals critically affects the scalability of RL training with more diverse motions, leading to a trade-off. While incorporating more expressive motion representations into the state space can accommodate finer-grained and more diverse motions, the expanded dimensionality will lead to a curse of sample efficiency in scalable RL. 2) Closing the embodiment gap: as evident in Section 8.5.1 and prior work [170], training on infeasible or damaged motions might largely harm performance. The feasibility of motions varies from robot to robot due to hardware constraints, and we lack systematic algorithms to identify feasible motions. We need more efforts to close this embodiment gap: on one end, more human-like humanoids would help; on the other, more teleoperation research is expected to improve the learnability of human motions. 3) Closing the sim-to-real gap: to achieve a successful sim-to-real transfer, regularization (e.g., reward regularization) and domain randomization are needed. However, over-regularization and over-randomization will also hinder the policy from learning the motions. It remains unknown how to strike the best trade-off between motion imitation leaning and sim-to-real transfer into a universal humanoid control policy.

Towards Real-time Humanoid Teleoperation. In this work, we leverage RGB and 3D pose estimator to transform the motions of human teleoperators into humanoid robots. The latency and error from RGB cameras and pose estimation also lead to an inevitable trade-off between efficiency and precision in teleoperation. Also, in this work, the human teleoperator receives feedback from the humanoid only in the form of visual perception. More research is needed on human-robot interaction to study this emerging multimodal interaction (e.g., force feedback [315], verbal and conversational feedback [37]), which could further enhance the capability of humanoid teleoperation.

Towards Whole-body Humanoid Teleoperation. One may wonder if lower-body tracking is necessary, as the major embodiment gap between humans and humanoids is the lower-body capability. A large proportion of skillful motions of humans (e.g., sports, dancing) need diverse agile lower-body movements. We emphasize the scenarios where legged robots hold an advantage over wheeled robots, in which lower-body tracking is necessary to follow human lower-body movements, including stepping stones, kicking, spread legs, etc. In the future, a teleoperated humanoid system that learns to switch between robust locomotion and skillful lower-body tracking would be a promising research direction.

8.7 Conclusions

In this study, we introduced Human to Humanoid (H2O), a scalable learning-based framework that enables real-time whole-body humanoid robot teleoperation using just an RGB camera. Our approach, leveraging reinforcement learning and a novel “sim-to-data" process, addresses the complex challenge of translating human motion into actions a humanoid robot can perform. Through comprehensive simulation and real-world tests, H2O demonstrated its capability to perform a wide range of dynamic tasks with high fidelity and minimal hardware requirements. In the next chapter, we will make this policy more robust, support dexterous hands, and enable autonomy.

Chapter 9

OmniH2O: Universal and Dexterous Human-to-Humanoid Whole-Body Teleoperation and Learning

Figure 9.1. (a) OmniH2O enables teleoperating a full-size humanoid robot (Unitree H1) to complete tasks that require both high-precision manipulation and locomotion. (b) OmniH2O also enables full autonomy through visual input, controlled by GPT-4o or a policy learned from teleoperated demonstrations. **Videos**: see the anonymous website https://anonymous-omni-h2o.github.io/

9.1 Introduction

Building upon our teleoperation whole-body controller in the Chapter 8, in this work, we propose OmniH2O, a learning-based system for whole-body humanoid teleoperation and autonomy. We propose a pipeline to train a robust whole-body motion imitation policy via teach-student distillation and identify key factors in obtaining a stable control policy that supports dexterous manipulation. For instance, we find these elements to be essential: *motion data distribution*, *reward designs*, and *state space design* and *history utilization*. The distribution of the motion imitation dataset needs to be biased toward standing and squatting to help the policy learn to stabilize the lower body during manipulation. Regularization rewards are used to shape the desired motion but need to be applied with a curriculum. The input history could replace the global linear velocity, an essential input in previous work [98] that requires Motion Capture (MoCap) to obtain. We also carefully design our control interface and choose the kinematic pose as an intermediate representation to bridge between human instructions and humanoid actuation. This interface makes our control framework compatible with many real-world input sources, such as VR, RGB cameras, and autonomous agents (GPT-4o). Powered by our robust control policy, we demonstrate teleoperating humanoids to perform various daily tasks (racket swinging, flower watering, brush writing, squatting and picking, boxing, basket delivery, *etc.*), as shown in Figure 9.1. Through teleoperation, we collect a dataset of our humanoid completing six tasks such as hammer catching, basket picking, *etc.*, annotated with paired first-person RGBD camera views, control input, and whole-body motor actions. Based on the dataset, we showcase training autonomous policies via imitation learning.

In conclusion, our contributions are as follows: **(1)** We propose a pipeline to train a robust humanoid control policy that supports whole-body dexterous loco-manipulation with a universal interface that enables versatile human control and autonomy. **(2)** Experiments of large-scale motion tracking in simulation and the real world validate the superior motion imitation capability of OmniH2O. **(3)** We contribute the first humanoid loco-manipulation dataset and evaluate imitation learning methods on it to demonstrate humanoid whole-body skill learning from teleoperated datasets.

9.2 Related Works

Learning-based Humanoid Control. Controlling humanoid robots is a long-standing robotic problem due to their high degree-of-freedom (DoF) and lack of self-stabilization [85, 102]. Recently, learning-based methods have shown promising results [43, 58, 68, 98, 147, 148, 223, 224, 252]. However, most studies [58, 68, 147, 148, 223, 224, 252] focus mainly on learning robust locomotion policies and do not fully unlock all the abilities of humanoids. For tasks that require whole-body loco-manipulation, the lower body must serve as the support for versatile and precise upper body movement [76]. Traditional goal-reaching [100, 317] or velocity-tracking objectives [43] used in legged locomotion are incompatible with such requirements because these objectives require additional task-specific lower-body goals (from other policies) to indirectly account for upper-lower-body coordination. OmniH2O instead learns an end-to-end whole-body policy to coordinate upper and lower bodies.

Humanoid Teleoperation. Teleoperating humanoids holds great potential in unlocking the full capabilities of the humanoid system. Prior efforts in humanoid teleoperation have used task-space control [55, 246], upper-body-retargeted teleoperation [36, 70] and whole-body teleoperation [98, 107, 115, 186, 198, 218, 246, 264]. Recently, H2O [98] presents an RL-based whole-body teleoperation framework that uses a third-person RGB camera to obtain full-body keypoints of the human teleoperator. However, due to the delay and inaccuracy of RGB-based pose estimation and the requirement for global linear velocity estimation, H2O [98] requires MoCap during test time, only supports simple mobility tasks, and lacks the precision for dexterous manipulation tasks. By contrast, OmniH2O enables high-precision dexterous loco-manipulation indoors and in the wild.

Whole-body Humanoid Control Interfaces. To control a full-sized humanoid, many interfaces such as exoskeleton [116], MoCap [56, 259], and VR [75, 103] are proposed. Recently, VR-based humanoid control [138, 170, 295, 319] has been drawing attention in the graphics community due to its ability to create whole-body motion using sparse input. However, these VR-based works only focus on humanoid control for animation and do not support mobile manipulation. OmniH2O, on the other hand, can control a real humanoid robot to complete real-world manipulation tasks.

Open-sourced Robotic Dataset and Imitation Learning. One major challenge within the robotics community is the limited number of publicly available datasets compared to those for language and vision tasks [199]. Recent efforts [27, 28, 66, 127, 151, 199, 271, 286] have focused on collecting robotic data using various embodiments for different tasks. However, most of these datasets are collected with fixed-base robotic arm platforms. Even one of the most comprehensive datasets to date, Open X-Embodiment [199], does not include data for humanoids. To the best of our knowledge, we are the first to release a dataset for full-sized humanoid whole-body loco-manipulation.

9.3 Universal and Dexterous Human-to-Humanoid Whole-Body Control

In this section, we describe our whole-body control system to support teleoperation, dexterous manipulation, and data collection. As simulation has access to inputs that are hard to obtain from real-world devices, we opt to use a teacher-student framework. We also provide details about key elements to obtain a stable and robust control policy: dataset balance, reward designs, *etc*.

Problem Formulation. We formulate the learning problem as goal-conditioned RL for a Markov Decision Process (MDP) defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ of state $\mathcal{S}$, action $\boldsymbol{a}_t \in \mathcal{A}$, transition $\mathcal{T}$, reward function $\mathcal{R}$, and discount factor γ. The state $\boldsymbol{s}_t$ contains the proprioception $\boldsymbol{s}_t^{\text{p}}$ and the goal state $\boldsymbol{s}_t^{\text{g}}$. The goal state $\boldsymbol{s}_t^{\text{g}}$ includes the motion goals from the human teleoperator or autonomous agents. Based on proprioception $\boldsymbol{s}_t^{\text{p}}$, goal state $\boldsymbol{s}_t^{\text{g}}$, and action $\boldsymbol{a}_t$, we define the reward $r_t = \mathcal{R}\left(\boldsymbol{s}_t^{\text{p}}, \boldsymbol{s}_t^{\text{g}}, \boldsymbol{a}_t\right)$. The action $\boldsymbol{a}_t$ specifies the target joint angles and a PD controller actuates the motors. We apply the Proximal Policy Optimization algorithm (PPO) [245] to maximize the cumulative discounted reward $\mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} r_t\right]$. In this work, we study the motion imitation task where our policy π_{OmniH2O} is trained to track real-time motion input as shown in Figure 9.3. This task provides a universal interface for humanoid control as the kinematic pose can be provided by many different sources. We define kinematic pose as $\boldsymbol{q}_t \triangleq (\boldsymbol{\theta}_t, \boldsymbol{p}_t)$, consisting of 3D joint rotations $\boldsymbol{\theta}_t$ and positions $\boldsymbol{p}_t$ of all joints on the humanoid. To define velocities $\dot{\boldsymbol{q}}_{1:T}$, we have $\dot{\boldsymbol{q}}_t \triangleq (\boldsymbol{\omega}_t, \boldsymbol{v}_t)$ as angular $\boldsymbol{\omega}_t$ and linear velocities $\boldsymbol{v}_t$. As a notation convention, we use $\tilde{\cdot}$ to represent kinematic quantities from VR headset or pose generators, $\hat{\cdot}$ to denote ground truth quantities from MoCap datasets, and normal symbols without accents for values from the physics simulation or real robot.

Human Motion Retargeting. We train our motion imitation policy using retargeted motions from the AMASS [176] dataset, using a similar retargeting process as H2O [98]. One major drawback of H2O is that the humanoid tends to take small adjustment steps instead of standing still. In order to enhance the ability of stable standing and squatting, we bias our training data by adding sequences that contain fixed lower body motion. Specifically, for each motion sequence $\hat{\boldsymbol{q}}_{1:T}$ from our dataset, we create a "stable" version $\hat{\boldsymbol{q}}_{1:T}^{\text{stable}}$ by fixing the root position and the lower body to a standing or squatting position as shown in Fig. 9.2. We provide

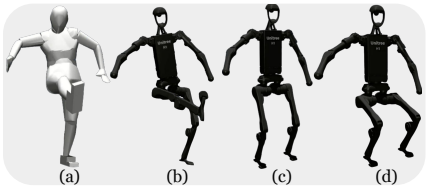


Figure 9.2. (a) Source motion; (b) Retargeted motion; (c) Standing variant; (d) Squatting variant.

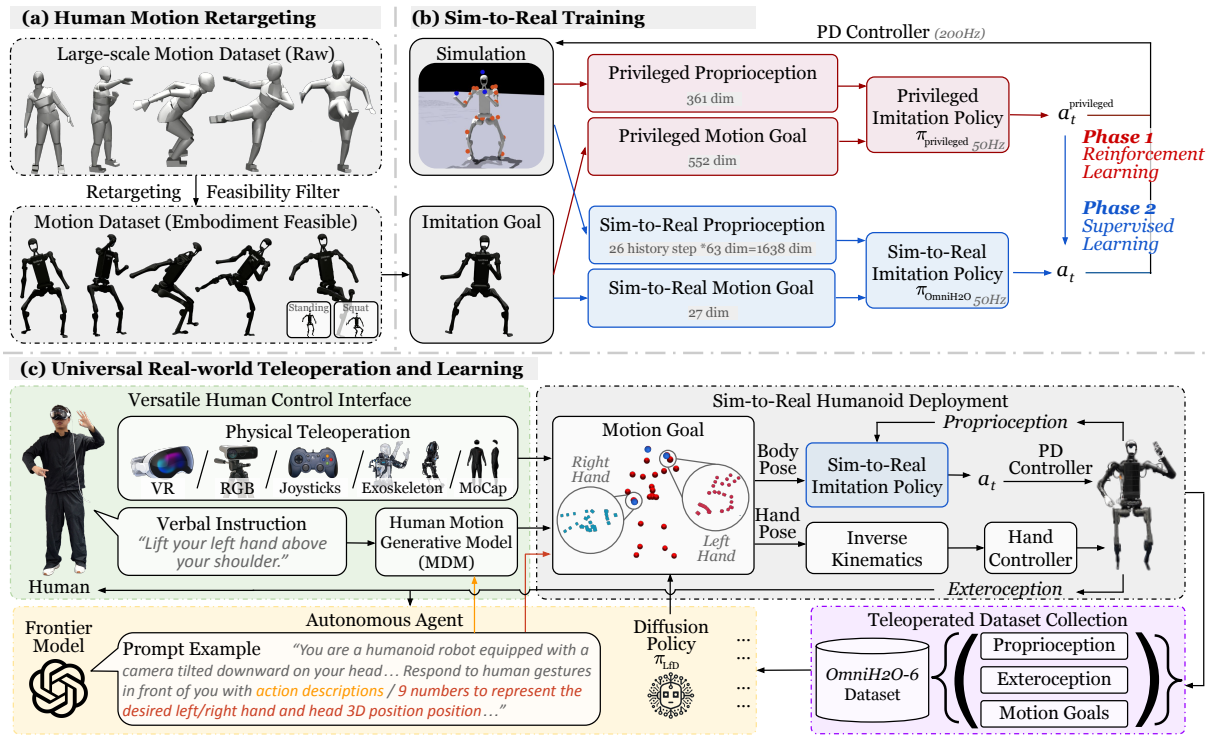


Figure 9.3. (a) OmniH2O retargets large-scale human motions and filters out infeasible motions for humanoids. (b) Our **sim-to-real policy** is distilled through supervised learning from an **RL-trained teacher policy** using privileged information. (c) The universal design of OmniH2O supports versatile **human control interfaces** including VR headset, RGB camera, language, *etc.* Our system also supports to be controlled by **autonomous agents** like GPT-4o or imitation learning policy trained using our **dataset collected via teleoperation**.

ablation of this strategy in Section 9.5.8.

Reward and Domain Randomization. To train $\pi_{\text{privileged}}$ that is suitable as a teacher for a real-world deployable student policy, we employ both imitation rewards and regularization rewards. Previous work [43, 98] often uses regularization rewards like *feet air time* or *feet height* to shape the lower-body motions. However, these rewards result in the humanoid stomping to keep balanced instead of standing still. To encourage standing still and taking large steps during locomotion, we propose a key reward function *max feet height for each step*. The *max feet height for each step* reward is designed to encourage the robot to take higher steps by rewarding it based on the maximum height achieved by the feet during the air phase of each step. We find that this reward, when applied with a carefully designed curriculum, effectively helps RL decide when to stand or walk. We provide a detailed overview of rewards, curriculum design, and domain randomization in Sections 9.5.5 and 9.5.6.

Teacher: Privileged Imitation Policy. During real-world teleoperation of a humanoid robot, much information that is accessible in simulation (*e.g.*, the global linear/angular velocity of every body link) is not available. Moreover, the input to a teleoperation system could be *sparse* (*e.g.*, for VR-based teleoperation, only the hands and head's poses are known), which makes the RL optimization challenging. To tackle this issue, We first train a teacher policy that uses privileged state information and then distill it to a student policy with limited state space. Having access to the privileged state can help RL find more optimal solutions, as shown in prior works [137] and our experiments (Section 9.4). Formally, we train a privileged motion imitator $\pi_{\text{privileged}}(\boldsymbol{a}_t | \boldsymbol{s}_t^{\text{p-privileged}}, \boldsymbol{s}_t^{\text{g-privileged}})$, as described in Figure 9.3. The

proprioception is defined as $s_t^{\text{p-privileged}} \triangleq [p_t, \theta_t, \dot{q}_t, \omega_t, a_{t-1}]$, which contains the humanoid rigidbody position p_t, orientation θ_t, linear velocity $\dot{q}_t$, angular velocity ω_t, and the previous action a_{t-1}. The goal state is defined as $s_t^{\text{g-privileged}} \triangleq [\hat{\theta}_{t+1} \ominus \theta_t, \hat{p}_{t+1} - p_t, \hat{v}_{t+1} - v_t, \hat{\omega}_{t+1} - \omega_t, \hat{\theta}_{t+1}, \hat{p}_{t+1}]$, which contains the reference pose $(\hat{\theta}_t, \hat{p}_t)$ and one-frame difference between the reference and current state for all rigid bodies of the humanoid.

Student: Sim-to-Real Imitation Policy with History. We design our control policy to be compatible with many input sources by using the kinematic reference motion as the intermediate representation. As estimating full-body motion $\tilde{q}_t$ (both rotation and translation) is difficult (especially from VR headsets), we opt to control our humanoid with position $\tilde{p}_t$ only for teleoperation. Specifically, for real-world teleoperation, the goal state is $s_t^{\text{g-real}} \triangleq (\tilde{p}_t^{\text{real}} - p_t^{\text{real}}, \tilde{v}_t^{\text{real}} - v_t^{\text{real}}, \tilde{p}_t^{\text{real}})$. The superscript $^{\text{real}}$ indicates using the 3-points available (head and hands) from the VR headset. For other control interfaces (e.g., RGB, language), we use the same input 3-point input to maintain consistency, though can be easily extended to more keypoints to alleviate ambiguity. For proprioception, the student policy $s_t^{\text{p-real}} \triangleq (d_{t-25:t}, \dot{d}_{t-25:t}, \omega_{t-25:t}^{\text{root}}, g_{t-25:t}, a_{t-25-1:t-1})$ uses values easily accessible in the real-world, which includes 25-step history of joint (DoF) position $d_{t-25:t}$, joint velocity $\dot{d}_{t-25:t}$, root angular velocity $\omega_{t-25:t}^{\text{root}}$, root gravity $g_{t-25:t}$, and previous actions $a_{t-25-1:t-1}$. The inclusion of history data helps improve the robustness of the policy with our teacher-student supervised learning. Note that no global linear velocity v_t information is included in our observations and the policy implicitly learns velocity using history information. This removes the need for MoCap as in H2O [98] and further enhances the feasibility of in-the-wild deployment.

Policy Distillation. We train our deployable teleoperation policy π_{OmniH2O} following the DAgger [239] framework: for each episode, we roll out the student policy $\pi_{\text{OmniH2O}}(a_t | s_t^{\text{p-real}}, s_t^{\text{g-real}})$ in simulation to obtain trajectories of $(s_{1:T}^{\text{p-real}}, s_{1:T}^{\text{g-real}})$. Using the reference pose $\hat{q}_{1:T}$ and simulated humanoid states $s_{1:T}^{\text{p}}$, we can compute the privileged states $s_t^{\text{g-privileged}}, s_t^{\text{p-privileged}} \leftarrow (s_t^{\text{p}}, \hat{q}_{t+1})$. Then, using the pair $(s_t^{\text{p-privileged}}, s_t^{\text{g-privileged}})$, we query the teacher $\pi_{\text{privileged}}(a_t{}^{\text{privileged}} | s_t^{\text{p-privileged}}, s_t^{\text{g-privileged}})$ to calculate the reference action $a_t{}^{\text{privileged}}$. To update π_{OmniH2O}, the loss is: $\mathcal{L} = \|a_t{}^{\text{privileged}} - a_t\|_2^2$.

Dexterous Hands Control. As shown in Figure 9.3(c), we use the hand poses estimated by VR [44, 203], and directly compute joint targets based on inverse kinematics for an off-the-shelf low-level hand controller. We use VR for the dexterous hand control in this work, but the hand pose estimation could be replaced by other interfaces (e.g., MoCap gloves [247] or RGB cameras [92]) as well.

9.4 Experimental Results

In our experiments, we aim to answer the following questions. **Q1.** (Section 9.4.1) Can OmniH2O accurately track motion in simulation and real world? **Q2.** (Section 9.4.2) Does OmniH2O support versatile control interfaces in the real world and unlock new capabilities of loco-manipulation? **Q3.** (Section 9.4.3) Can we use OmniH2O to collect data and learn autonomous agents from teleoperated demonstrations? As motion is best seen in videos, we provide visual evaluations in our `website`.

9.4.1 Whole-body Motion Tracking

Experiment Setup. To answer **Q1**, we evaluate OmniH2O on motion tracking in simulation (Section 9.4.1) and the real world (Section 9.4.1). In simulation, we evaluate on the retargeted AMASS dataset with augmented motions $\hat{Q}$ (14k sequences); in real-world, we test on 20 standing sequences due to the limited physical lab space and the difficulty of evaluating on large-scale datasets in the real world. Detailed state-space composition (Section 9.5.3), ablation setup (Section 9.5.2), hyperparameters (Table 9.18), and hardware configuration (Section 9.5.1) are summarized in the Appendix.

			All sequences					Successful sequences			
Method	State Dimension	Sim2Real	Succ ↑	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓	$E_{\text{g-mpjpe}}$ ↓	E_{mpjpe} ↓	E_{acc} ↓	E_{vel} ↓
Privileged policy	$\mathcal{S} \subset \mathcal{R}^{913}$	✗	94.77%	126.51	70.68	3.57	6.20	122.71	69.06	2.22	5.20
H2O [98]	$\mathcal{S} \subset \mathcal{R}^{138}$	✓	87.52%	148.13	81.06	5.12	7.89	133.28	75.99	2.40	5.75
OmniH2O	$\mathcal{S} \subset \mathcal{R}^{1665}$	✓	**94.10%**	**141.11**	**77.82**	**3.70**	**6.54**	**135.49**	**75.75**	**2.30**	**5.47**
(a) Ablation on DAgger/RL											
OmniH2O-w/o-DAgger-History0	$\mathcal{S} \subset \mathcal{R}^{90}$	✗	90.62%	163.44	91.29	5.12	8.80	153.31	87.59	3.15	7.27
OmniH2O-w/o-DAgger	$\mathcal{S} \subset \mathcal{R}^{1665}$	✗	47.11%	223.27	128.90	15.03	16.29	182.13	119.54	5.47	9.10
OmniH2O-History0	$\mathcal{S} \subset \mathcal{R}^{90}$	✓	93.80%	141.21	78.52	3.74	6.62	**134.90**	76.11	**2.25**	5.48
OmniH2O	$\mathcal{S} \subset \mathcal{R}^{1665}$	✓	**94.10%**	**141.11**	**77.82**	**3.70**	**6.54**	135.49	**75.75**	2.30	**5.47**
(b) Ablation on History steps/Architecture											
OmniH2O-History50	$\mathcal{S} \subset \mathcal{R}^{3240}$	✓	93.56%	141.51	78.51	4.01	6.79	135.04	76.07	2.36	5.55
OmniH2O-History5	$\mathcal{S} \subset \mathcal{R}^{405}$	✓	93.60%	**139.23**	**77.82**	3.91	6.66	132.67	75.33	2.24	5.41
OmniH2O-History0	$\mathcal{S} \subset \mathcal{R}^{90}$	✓	93.80%	141.21	78.52	3.74	6.62	**134.90**	76.11	**2.25**	5.48
OmniH2O-GRU	$\mathcal{S} \subset \mathcal{R}^{90}$	✓	92.85%	147.67	80.84	4.05	6.93	142.75	79.10	2.38	5.66
OmniH2O-LSTM	$\mathcal{S} \subset \mathcal{R}^{90}$	✓	91.03%	147.36	80.34	4.12	7.04	142.64	78.59	2.37	5.72
OmniH2O-History25 (Ours)	$\mathcal{S} \subset \mathcal{R}^{1665}$	✓	**94.10%**	141.11	**77.82**	**3.70**	**6.54**	135.49	**75.75**	2.30	**5.47**
(c) Ablation on Tracking Points											
OmniH2O-22points	$\mathcal{S} \subset \mathcal{R}^{1836}$	✓	**94.72%**	**127.71**	**70.39**	**3.62**	**6.25**	**123.87**	**68.92**	**2.22**	**5.24**
OmniH2O-8points	$\mathcal{S} \subset \mathcal{R}^{1710}$	✓	94.31%	129.30	71.70	3.78	6.39	125.14	70.07	2.22	5.26
OmniH2O-3points (Ours)	$\mathcal{S} \subset \mathcal{R}^{1665}$	✓	94.10%	141.11	77.82	3.70	6.54	135.49	75.75	2.30	5.47
(d) Ablation on Linear Velocity											
OmniH2O-w-linvel	$\mathcal{S} \subset \mathcal{R}^{1743}$	✓	93.80%	**138.18**	78.12	3.94	6.61	**132.44**	75.98	**2.29**	**5.40**
OmniH2O	$\mathcal{S} \subset \mathcal{R}^{1665}$	✓	**94.10%**	141.11	**77.82**	**3.70**	**6.54**	135.49	**75.75**	2.30	5.47

Table 9.1. Simulation motion imitation evaluation of OmniH2O and baselines on dataset $\hat{\boldsymbol{Q}}$. Note that all the variants are trained with exact same rewards, domain randomizations and motion dataset $\hat{\boldsymbol{Q}}$.

Metrics. We evaluate the motion tracking performance using both pose and physics-based metrics. We report Success rate (Succ) as in PHC [169], where imitation is unsuccessful if the average deviation from reference is farther than 0.5m at any point in time. Succ measures whether the humanoid can track the reference motion without losing balance or lagging behind. The global MPJPE $E_{g-\text{mpjpe}}$ and the root-relative mean per-joint position error (MPJPE) E_{mpjpe} (in mm) measures our policy's ability to imitate the reference motion globally and locally (root-relative). To show physical realism, we report average joint acceleration E_{acc} (mm/frame^2) and velocity E_{vel} (mm/frame) error.

Simulation Motion-Tracking Results

In Table 9.1's first three rows, we can see that our deployable student policy significantly improves upon prior art [98] on motion imitation and achieves a similar success rate as the teacher policy.

Ablation on DAgger/RL. We test the performance of OmniH2O without DAgger (*i.e.*, directly using RL to train the student policy). In Table 9.1(a) we can see that DAgger improves performance overall, especially for policy with history input. Without DAgger the policy struggles to learn a coherent policy when provided with a long history. This is due to RL being unable to handle the exponential growth in input complexity. However, the history information is necessary for learning a deployable policy in the real-world, providing robustness and implicit global velocity information (see Section 9.4.1). Supervised learning via DAgger is able to effectively leverage the history input and is able to achieve better performance.

Ablation on History Steps/Architecture. In Table 9.1(b), we experiment with varying history steps (0, 5, 25, 50) and find that 25 steps achieve the best balance between performance and learning efficiency. Additionally, we evaluate different neural network architectures for history utilization: MLP, LSTM, GRU and determine that MLP-based OmniH2O performs the best.

Ablation on Sparse Input. To support VR-based teleoperation, π_{OmniH2O} only tracks 3-points (head

		Tested sequences			
Method	State Dimensions	$E_{\text{g-mpjpe}} \downarrow$	$E_{\text{mpjpe}} \downarrow$	$\text{E}_{\text{acc}} \downarrow$	$\text{E}_{\text{vel}} \downarrow$
H2O [98]	$\mathcal{S} \subset \mathcal{R}^{138}$	87.33	53.32	6.03	5.87
OmniH2O	$\mathcal{S} \subset \mathcal{R}^{1665}$	**47.94**	**41.87**	**1.84**	**2.20**
(a) Ablation on Real-world Linear Velocity Estimation					
OmniH2O-w-linvel(VIO)[1,2]	$\mathcal{S} \subset \mathcal{R}^{1743}$	N/A	N/A	N/A	N/A
OmniH2O-w-linvel(MLP)	$\mathcal{S} \subset \mathcal{R}^{1743}$	50.93	42.47	2.16	2.26
OmniH2O-w-linvel(GRU)	$\mathcal{S} \subset \mathcal{R}^{1743}$	49.75	42.38	2.20	2.31
OmniH2O	$\mathcal{S} \subset \mathcal{R}^{1665}$	**47.94**	**41.87**	**1.84**	**2.20**
(b) Ablation on History steps/Architecture					
OmniH2O-History0	$\mathcal{S} \subset \mathcal{R}^{90}$	83.26	46.00	4.86	4.45
OmniH2O-History5	$\mathcal{S} \subset \mathcal{R}^{405}$	62.18	46.50	2.66	2.90
OmniH2O-History50	$\mathcal{S} \subset \mathcal{R}^{3240}$	50.24	**40.11**	2.37	2.71
OmniH2O-LSTM	$\mathcal{S} \subset \mathcal{R}^{90}$	87.00	46.06	3.89	3.88
OmniH2O	$\mathcal{S} \subset \mathcal{R}^{1665}$	**47.94**	41.87	**1.84**	**2.20**

[1] Use ZED SDK to estimate the linear velocity.

[2] Unable to finish the real-world test due to falling on the ground.

Table 9.2. Real-world motion tracking evaluation on 20 standing motions in $\hat{\boldsymbol{Q}}$

and hands) to produce whole-body motion. The impact of the number of tracking points is examined in Table 9.1(c). We test configurations ranging from minimal (3) to full-body motion goal (22) and found that 3-point tracking can achieve comparable performance with more input keypoints. As expected, 3-point policy sacrifices some whole-body motion tracking accuracy but gains greater applicability to commercially available devices.

Ablation on Global Linear Velocity. Given the challenges associated with global velocity estimation in real-world applications, we compare policies trained with and without explicit velocity information. In Table 9.1(d), we find that linear velocity information does not boost performance in simulation, but it introduces significant challenges in real-world deployment (details illustrated in Section 9.4.1), prompting us to develop a policy with state spaces that do not depend on linear velocity as proprioception to avoid these issues.

Real-world Motion-Tracking Results

Ablation on Real-world Linear Velocity Estimation. We exclude linear velocity in our state space design global linear velocity obtained by algorithms such as visual inertial odometry (VIO) can be rather noisy, as shown in Section 9.5.7. Our ablation study (Table 9.2(a)) also shows that policies without velocity input has better performance compared with policies using velocities estimated by VIO or MLP/GRU neural estimators (implementation details in Section 9.5.7), which suggests that the policy with history can effectively track motions without explicit linear velocity as input.

History Steps and Architecture. Real-world evaluation in Table 9.2(b) also shows that our choice of 25 steps of history achieves the best performance. The tracking performance of LSTM shows that MLP-based policy performs better in the real-world.

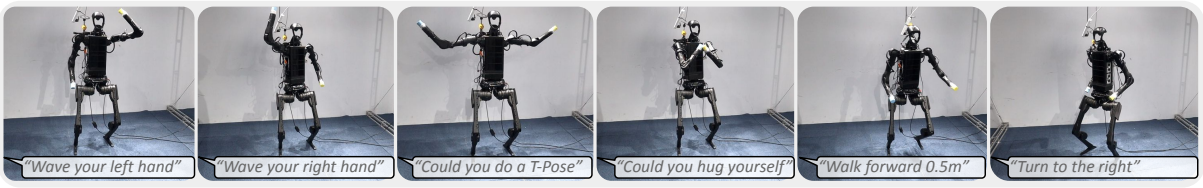


Figure 9.4. OmniH2O policy tracks motion goals from a language-based human motion generative model [277].

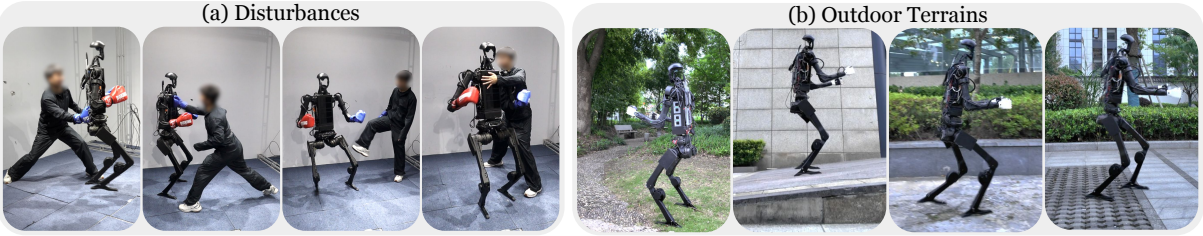


Figure 9.5. OmniH2O shows superior robustness against human strikes and different outdoor terrains.

9.4.2 Human Control via Universal Interfaces

To answer **Q2**, we demonstrate real-world capabilities of OmniH2O with versatile human control interfaces. All the capabilities discussed below utilize the same motion-tracking policy π_{OmniH2O}.

Teleoperation. We teleoperate the humanoid using π_{OmniH2O} with both VR and RGB camera as interfaces. The results are shown in Figure 9.1(a) and Section 9.5.9, where the robot is able to finish dexterous loco-manipulation tasks with high precision and robustness.

Language Instruction Control. By linking π_{OmniH2O} with a pretrained text to motion generative model (MDM) [277], it enables controlling the humanoid via verbal instructions. As shown in Figure 9.4, with humans describing desired motions, such as "*raise your right hand*". MDM generates the corresponding motion goals that are tracked by the OmniH2O.

Robustness Test. As shown in Figure 9.5, we test the robustness of our control policy. We use the same policy π_{OmniH2O} across all tests, whether with fixed standing motion goals or motion goals controlled by joysticks, either moving forward or backward. With human punching and kicking from various angles, the robot, without external assistance, is able to maintain stability on its own. We also test OmniH2O on various outdoor terrains, including grass, slopes, gravel, *etc*. OmniH2O demonstrates great robustness under disturbances and unstructured terrains.

9.4.3 Autonomy via Frontier Models or Imitation Learning

To answer **Q3**, we need to bridge the whole-body tracking policy (*physical intelligence*), with automated generation of kinematic motion goals through visual input (*semantic intelligence*). We explore two ways of automating humanoid control with OmniH2O: (1) using multi-modal frontier models to generate motion goals and (2) learning autonomous policies from the teleoperated dataset.

GPT-4o Autonomous Control. We integrate our system, OmniH2O, with GPT-4o, utilizing a head-mounted camera on the humanoid to capture images for GPT-4o (Figure 9.6). The prompt (details in Section 9.5.13) provided to GPT-4o offers several motion primitives for it to choose from, based on the current visual context. We opt for motion primitives rather than directly generating motion goals because of GPT-4o's relatively long response time. As shown in Figure 9.6, the robot manages to give the correct punch based on the color of the target and successfully greets a human based on the intention indicated

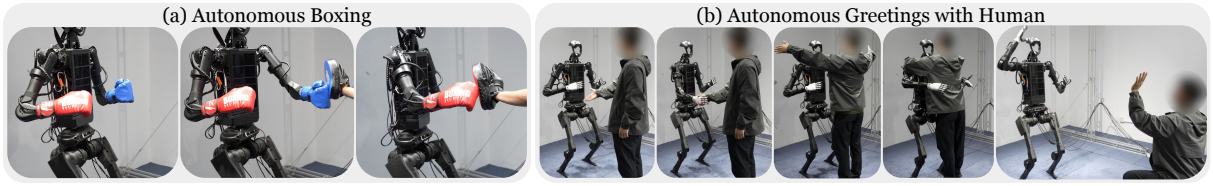


Figure 9.6. OmniH2O sends egocentric RGB views to GPT-4o and executes the selected motion primitives.

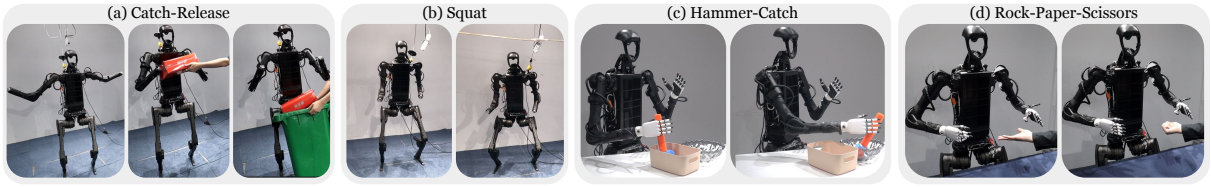


Figure 9.7. OmniH2O autonomously conducts four tasks using LfD models trained with our collected data.

by human poses.

***OmniH2O-6* Dataset**. We collect demonstration data via VR-based teleoperation. We consider six tasks: Catch-Release, Squat, Rope-Paper-Scissors, Hammer-Catch, Boxing, and Pasket-Pick-Place. Our dataset includes paired RGBD images from the head-mounted camera, the motion goals of H1's head and hands with respect to the root, and joint targets for motor actuation, recorded at 30Hz. For simple tasks such as Catch-Release, Squat, and Rope-Paper-Scissors, approximately 5 minutes of data are recorded, and for tasks like Hammer-Catch and Basket-Pick-Place, we collect approximately 10 minutes, leading to 40-min real-world humanoid teleoperated demonstrations in total. Detailed task descriptions of the six open-sourced datasets are in Appendix 9.5.10.

Humanoid Learning from Demonstrations. We design our learning from demonstration policy to be $\pi_{\text{LfD}}(\hat{\boldsymbol{p}}^{\text{Sparse-lfd}}_{t:t+\phi}|\boldsymbol{I}_t)$, where π_{LfD} outputs ϕ frames of motion goals given the image input $\boldsymbol{I}_t$. Here, we also include dexterous hand commands in $\hat{\boldsymbol{p}}^{\text{Sparse-lfd}}_{t:t+\phi}$. Then, our $\pi_{\text{OmniH2O}}(\boldsymbol{a}_t|\boldsymbol{s}^{\text{p-real}}_t, \boldsymbol{s}^{\text{g-real}}_t)$ serves as the low-level policy to compute joint actuations for humanoid whole-body control. The training hyperparameters are in Section 9.5.12. Compared to directly using the π_{LfD} to output joint actuation, we leverage the trained motor skills in π_{OmniH2O}, which drastically reduces the number of demonstrations needed. We benchmark a variety of imitation learning algorithms on four tasks in our collected dataset (shown in Figure 9.7), including Diffusion Policy [47] with Denoising Diffusion Probabilistic Model [104] (DP-DDPM) and Denoising Diffusion Implicit Model [258] (DP-DDIM) and vanilla Behavior Cloning with a ResNet architecture (BC). Detailed descriptions of these methods are provided in Section 9.5.4.

Metrics	All Tasks		
(a) Ablation on Data size			
	25%data	50%data	100%data
MSE Loss	1.30E-2	7.48E-3	5.25E-4
Succ rate	4/10	6.5/10	8/10
(b) Ablation on Sequence observation/action			
	Si-O-Si-A	Se-O-Se-A	Si-O-Se-A
MSE Loss	4.89E-4	9.91E-4	5.25E-4
Succ rate	6.5/10	8.75/10	8/10
(c) Ablation on BC/DDIM/DDPM			
	BC	DP-DDIM	DP-DDPM
MSE Loss	5.63E-3	1.9E-3	5.25E-4
Succ rate	1/10	7.75/10	8/10

Table 9.3. Quantitative LfD average performance on 4 tasks over 10 runs.

To evaluate π_{LfD}, we report the average MSE loss and the success rate in Table 9.3, where we average the metrics across all tasks. More details for each task evaluation can be found in Section 9.5.10. We draw two key conclusions: (1) The Diffusion Policy significantly outperforms vanilla BC with ResNet; (2) In our LfD training, predicting a sequence of actions is crucial, as it enables the robot to effectively learn and replicate the trajectory.

9.5 Additional Details

9.5.1 Real Robot System Setup

Our real robot employs a Unitree H1 platform [282], outfitted with Damiao DM-J4310-2EC motors [57] and Inspire hands [112] for its manipulative capabilities. We have two versions of real robot computing setup. (1) The first one has two 16GB Orin NX computers mounted on the back of the H1 robot. The first Orin NX is connected to a ZED camera mounted on the waist of H1, which performs computations to determine H1's own location for positioning. The camera operates at 60 Hz FPS. Additionally, this Orin NX connects via Wi-Fi to our Vision Pro device to continuously receive motion goal information from a human operator. The second Orin NX serves as our main control hub. It receives the motion goal information, which it uses as input for our control policy. This policy then outputs torque information for each of the robot's motors and sends these commands to the robot. As control for the robot's fingers and wrists does not require inference, it is directly mapped from the Vision Pro data to the corresponding joints on the robot. The policy's computation frequency is set at 50 Hz. The two Orin NX units are connected via Ethernet, sharing information through a common ROS (Robot Operating System) network. The final commands to H1 are consolidated and dispatched by the second Orin NX. Our entire system has a low latency of only 20 milliseconds. It's worth noting that we designed the system in this way partly because the ZED camera requires substantial computational resources. By dedicating the first Orin NX to the ZED camera, and the second to policy inference, we ensure that each component operates with optimal performance. (2) In the second setup, a laptop (13th Gen i9-13900HX and NVIDIA RTX4090, 32GB RAM) serves as the computing and communication device. All devices, including the ZED camera, control policy, and Vision Pro, communicate through this laptop on its ROS system, facilitating centralized data handling and command dispatch. These two setups yield similar performance, and we use them interchangeably in our experiments.

9.5.2 Simulation Baseline and Ablations

In this section, we provide an explanation of each ablation method. **Main results in Table 9.1**

- **Privileged policy**: This teacher policy $\pi_{\text{privileged}}$ incorporates all privileged environment information, along with complete motion goal and proprioception data in the observations. State space composition details in Section 9.5.3.
- **H2O**: A policy trained using RL without DAgger and historical data, utilizing 8 keypoints of motion goal in observations. State space composition details in Section 9.5.3.
- **OmniH2O**: Our deployment policy π_{OmniH2O} that includes historical information and uses 3 keypoints of motion goal in observations, trained with DAgger. State space composition details in Table 9.6.

Ablation on DAgger/RL in Table 9.1(a)

- **OmniH2O-w/o-DAgger-History0**: This variant of OmniH2O is trained solely using RL and does not incorporate historical information within observations. State space composition details in Table 9.7.
- **OmniH2O-w/o-DAgger**: This model is trained using RL, excludes DAgger, but includes historical information from the last 25 steps in observations. State space composition details in Table 9.8.
- **OmniH2O-History0**: This model is trained with DAgger, but excludes historical information from the last 25 steps in observations. State space composition details in Table 9.9.

- **OmniH2O**: This model is trained with DAgger and incorporates 25-step historical information within observations. State space composition details in Table 9.6.

Ablation on History steps/Architecture in Table 9.1(b)

- **OmniH2O-History50/25/5/0**: This variant of the OmniH2O with 50, 25, or 0 steps of historical information in the observations. State space composition details in Table 9.10.
- **OmniH2O-GRU/LSTM**: This version replaces the MLP in the policy network with either GRU or LSTM, inherently incorporating historical observations. State space composition details in Table 9.11.

Ablation on Tracking Points in Table 9.1(c)

- **OmniH2O-22/8/3points**: This variant of the OmniH2O policy includes 22, 8, or 3 keypoints of motion goal in the observations, with the 3 keypoints setting corresponding to the standard OmniH2O policy. State space composition details in Tables 9.6, 9.12 and 9.13.

Ablation on Linear Velocity in Table 9.1(d)

- **OmniH2O-w-linvel**: This variant is almost the same as OmniH2O but with root linear velocity in observations and past linear velocity in history information. State space composition details in Table 9.14.

9.5.3 State Space Compositions

In this section, we introduce the detailed state space composition of baselines in the experiments.

Privileged Policy. This policy $\pi_{\text{privileged}}$ is the teacher policy that has access to all the available states for motion imitation, trained using RL.

State term	Dimensions
(Proprioception) Rigid Body position	66
(Proprioception) Rigid Body rotation	138
(Proprioception) Rigid Body velocity	69
(Proprioception) Rigid Body angular velocity	69
(Motion goal) Rigid Body position difference	69
(Motion goal) Rigid Body rotation difference	138
(Motion goal) Rigid Body velocity difference	69
(Motion goal) Rigid Body angular velocity difference	69
(Motion goal) Local Rigid Body position	69
(Motion goal) Local Rigid Body rotation	138
Actions	19
Total dim	913

Table 9.4. State space information in Privileged Policy setting

H2O. This policy has 8 keypoints input (shoulder, elbow, hand, leg) and with global linear velocity, trained using RL.

OmniH2O. This is our deployment policy π_{OmniH2O} with 25 history steps and without global linear velocity, trained using DAgger.

OmniH2O-w/o-DAgger-History0. This policy has a history of 0 steps, trained using RL.

OmniH2O-w/o-DAgger. This policy has the same architecture as OmniH2O but trained with RL.

State term	Dimensions
DoF position	19
DoF velocity	19
Base velocity	3
Base angular velocity	3
Base gravity	3
Motion goal	72
Actions	19
Total dim	138

Table 9.5. State space information in H2O setting

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Single step total dim	90
History state term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Actions	19
History Single step total dim	63
Total dim	1665(63*25 + 90)

Table 9.6. State space information in OmniH2O setting

OmniH2O-History0. This policy has a history of 0 steps, trained using DAgger.

OmniH2O-History*x*. This policy has a history of x steps, trained using DAgger.

OmniH2O-GRU/LSTM. This policy uses GRU/LSTM-based architecture, trained using DAgger.

OmniH2O-22points. This policy has 22 keypoints input (every joint on the humanoid), trained using DAgger.

OmniH2O-8points. This policy has 8 keypoints input (shoulder, elbow, hand, leg), trained using DAgger.

OmniH2O-w-linvel. This policy has 25 history steps and global linear velocity, trained using DAgger.

9.5.4 LfD Baselines

We conduct numerous ablation studies on LfD, aiming to benchmark the impact of various aspects on LfD tasks. The details of each ablation are as follows:

Ablation on Dataset size.

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Total dim	90

Table 9.7. State space information in OmniH2O-w/o-DAgger-History0 setting

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Single step total dim	90
History state term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Actions	19
History Single step total dim	63
Total dim	1665(63*25 + 90)

Table 9.8. State space information in OmniH2O-w/o-DAgger setting

- 25/50/100% data: In this task, we use 25/50/100% of the dataset as the training set. The algorithm is DDPM which takes a single-step image as input and outputs 8 steps of actions.

Ablation on Single/Sequence observation/action input/output.

- Si-O-Si-A: Single-step observation and single-step action mean that we take 1 step of image data as input and predict 1 step of action as output.
- Se-O-Se-A: Sequence-steps observation and sequence-steps actions mean that we take 4 steps of image data as input and predict 8 steps of action as output.
- Si-O-Se-A: Single-step observation and sequence-steps actions mean that we take 1 step of image data as input and predict 8 steps of action as output.

Ablation on Training Architecture..

- BC: Behavior cloning which means we use resnet+MLP to predict the next 8 steps action from the current step's image.

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Total dim	90

Table 9.9. State space information in OmniH2O-History0 setting

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Single step total dim	90
History state term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Actions	19
History Single step total dim	63
Total dim	63*x + 90

Table 9.10. State space information in OmniH2O-History$x$ setting

- DP-DDIM: We use DDIM as the algorithm which takes a single-step image as input and outputs 8 steps of actions.
- DP-DDPM: We use DDPM as the algorithm which takes a single-step image as input and outputs 8 steps of actions.

9.5.5 Reward Functions

Reward Components. Detailed reward components are summarized in Table 9.15.

Reward Curriculum. We have modified the cumulative discounted reward expression to handle multiple small rewards at each time step differently, depending on their sign. The revised formula is given by: $\mathbb{E}\left[\sum_{t=1}^{T}\gamma^{t-1}\sum_i s_{t,i}r_{t,i}\right]$ where $r_{t,i}$ represents different reward functions at time t, and $s_{t,i}$ is the scaling factor for each reward, defined as: $s_{t,i} = \begin{cases} s_{\text{current}} & \text{if } r_{t,i} < 0 \\ 1 & \text{if } r_{t,i} \geq 0 \end{cases}$ where $s_{current}$ is the scaling factor. This scaling factor is adjusted dynamically: it is multiplied by 0.9999 when the average episode length is less than 40, and multiplied by 1.0001 when it exceeds 120. The init $s_{current}$ is set to 0.5, then the upper bound

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Total dim	90

Table 9.11. State space information in OmniH2O-GRU/LSTM setting

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	198
Actions	19
Single step total dim	261
History state term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Actions	19
History Single step total dim	63
Total dim	1836(63*25+261)

Table 9.12. State space information in OmniH2O-22points setting

of this scaling factor is set to 1. This modification allows our policy to progressively learn from simpler to more complex scenarios with higher penalties, thereby reducing the difficulty for RL in exploring the optimal policy.

9.5.6 Domain Randomizations

Detailed domain randomization setups are summarized in Table 9.16.

9.5.7 Linear Velocity Estimation

The illustration of using the ZED camera VIO module and the comparison of VIO with neural state estimators are shown in Figure 9.8. We train our neural velocity estimators using a supervised learning approach. The process involves repeatedly deploying our policy in simulation with different motion goals. In every environment step, we use the root linear velocity to supervise our velocity estimator.

State term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Motion goal	72
Actions	19
Single step total dim	135
History state term	Dimensions
DoF position	19
DoF velocity	19
Base angular velocity	3
Base gravity	3
Actions	19
History Single step total dim	63
Total dim	1710(63*25+135)

Table 9.13. State space information in OmniH2O-8points setting

State term	Dimensions
DoF position	19
DoF velocity	19
Base velocity	3
Base angular velocity	3
Base gravity	3
Motion goal	27
Actions	19
Single step total dim	93
History state term	Dimensions
DoF position	19
DoF velocity	19
Base velocity	3
Base angular velocity	3
Base gravity	3
Actions	19
History Single step total dim	66
Total dim	1743(66*25 + 93)

Table 9.14. State space information in OmniH2O-w-linvel setting

9.5.8 Ablation on Dataset Motion Distribution

The ablation study on motion data distribution is shown in Figure 9.9. The policy trained without motion data augmentation is hard to stand still and make upper-body moves.

Term	Expression	Weight
	Penalty	
Torque limits	$\mathbb{1}(\boldsymbol{\tau}_t \notin [\boldsymbol{\tau}_{\min}, \boldsymbol{\tau}_{\max}])$	-2
DoF position limits	$\mathbb{1}(\boldsymbol{q}_t \notin [\boldsymbol{q}_{\min}, \boldsymbol{q}_{\max}])$	-125
DoF velocity limits	$\mathbb{1}(\dot{\boldsymbol{q}}_t \notin [\dot{\boldsymbol{q}}_{\min}, \dot{\boldsymbol{q}}_{\max}])$	-50
Termination	$\mathbb{1}_{\text{termination}}$	-250
	Regularization	
DoF acceleration	$\|\ddot{\boldsymbol{q}}_t\|_E 2$	-0.000011
DoF velocity	$\|\dot{\boldsymbol{q}}_t\|_2^2$	-0.004
Lower-body action rate	$\|\boldsymbol{a}_t^{\text{lower}} - \boldsymbol{a}_{t-1}^{\text{lower}}\|_2^2$	-3
Upper-body action rate	$\|\boldsymbol{a}_t^{\text{upper}} - \boldsymbol{a}_{t-1}^{\text{upper}}\|_2^2$	-0.625
Torque	$\|\boldsymbol{\tau}_t\|$	-0.0001
Feet air time	$T_{\text{air}} - 0.25$ [240]	1000
Max feet height for each step	$\max\{\text{max feet height for each step} - 0.25, 0\}$	1000
Feet contact force	$\|F_{\text{feet}}\|_2^2$	-0.75
Stumble	$\mathbb{1}(F_{\text{feet}}^{xy} > 5 \times F_{\text{feet}}^{z})$	-0.00125
Slippage	$\|\boldsymbol{v}_t^{\text{feet}}\|_2^2 \times \mathbb{1}(F_{\text{feet}} \geq 1)$	-37.5
Feet orientation	$\|\boldsymbol{g}_z^{\text{feet}}\|$	-62.5
In the air	$\mathbb{1}(F_{\text{feet}}^{\text{left}}, F_{\text{feet}}^{\text{right}} < 1)$	-200
Orientation	$\|\boldsymbol{g}_z^{\text{root}}\|$	-200
	Task Reward	
DoF position	$\exp(-0.25\|\hat{\boldsymbol{q}}_t - \boldsymbol{q}_t\|_2)$	32
DoF velocity	$\exp(-0.25\|\hat{\dot{\boldsymbol{q}}}_t - \dot{\boldsymbol{q}}_t\|_2^2)$	16
Body position	$\exp(-0.5\|\boldsymbol{p}_t - \hat{\boldsymbol{p}}_t\|_2^2)$	30
Body position VRpoints	$\exp(-0.5\|\boldsymbol{p}_t^{\text{real}} - \hat{\boldsymbol{p}}_t^{\text{real}}\|_2^2)$	50
Body rotation	$\exp(-0.1\|\boldsymbol{\theta}_t \ominus \hat{\boldsymbol{\theta}}_t\|)$	20
Body velocity	$\exp(-10.0\|\boldsymbol{v}_t - \hat{\boldsymbol{v}}_t\|_2)$	8
Body angular velocity	$\exp(-0.01\|\boldsymbol{\omega}_t - \hat{\boldsymbol{\omega}}_t\|_2)$	8

Table 9.15. Reward components and weights: penalty rewards for preventing undesired behaviors for sim-to-real transfer, regularization to refine motion, and task reward to achieve successful whole-body tracking in real-time.

9.5.9 Additional Physical Teleoperation Results

Additional VR-based and RGB-based teleoperation demo are shown in Figure 9.10.

9.5.10 Dataset and Imitation Learning

As shown in Figure 9.11, we collected 6 LfD tasks' dataset to enable the robot to autonomously perform certain functions.

Catch-Release: Catch a red box and release it into a trash bin. This task has 13234 frames in total.

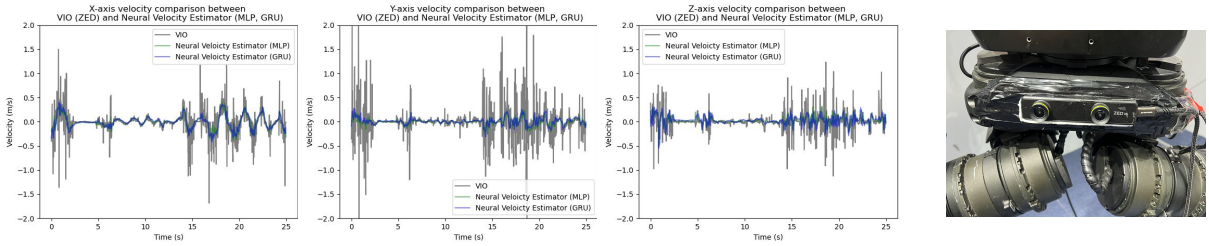
Squat: Squat when the robot sees a horizontal bar approaching that is lower than its head height. This task has 8535 frames in total.

Hammer-Catch: Use right hand to catch a hammer in a box. This task has 12759 frames in total.

Rock-Paper-Scissors: When the robot sees the person opposite it makes one of the rock-paper-scissors

Term	Value
Dynamics Randomization	
Friction	$\mathcal{U}(0.2, 1.1)$
Base CoM offset	$\mathcal{U}(-0.1, 0.1)$m
Link mass	$\mathcal{U}(0.7, 1.3) \times$ default kg
P Gain	$\mathcal{U}(0.75, 1.25) \times$ default
D Gain	$\mathcal{U}(0.75, 1.25) \times$ default
Torque RFI [32]	$0.1 \times$ torque limit N $\cdot$ m
Control delay	$\mathcal{U}(20, 60)$ms
Motion reference offset	$\mathcal{U}([-0.02, 0.02], [-0.02, 0.02], [-0.1, 0.1])$cm
External Perturbation	
Push robot	interval $= 5s$, $v_{xy} = 1$m/s
Randomized Terrain	
Terrain type	flat, rough, low obstacles [100]

Table 9.16. Here we describe the range of dynamics randomization for simulated dynamics randomization, external perturbation, and terrain, which are important for sim-to-real transfer, robustness, and generalizability.



(a) X-axis velocity (b) Y-axis velocity (c) Z-axis velocity (d) VIO setup

Figure 9.8. The illustration of using ZED camera VIO module, and the comparison of the velocity estimation of VIO with neural state estimators.

gestures, it should respond with the corresponding gesture that wins. This task has 9380 frames in total.
Boxing: When you see a blue boxing target, throw a left punch; when you see a red one, throw a right punch. This task has 11118 frames in total.
Basket-Pick-Place: Use your right hand to pick up the box and place it in the middle when the box is on the right side, and use your left hand if the box is on the left side. If you pick up the box with your right hand, place it on the left side using your left hand; if picked up with your left hand, place it on the right side using your right hand. This task has 18436 frames in total.

Figure 9.9. The ablation of data augmentation.

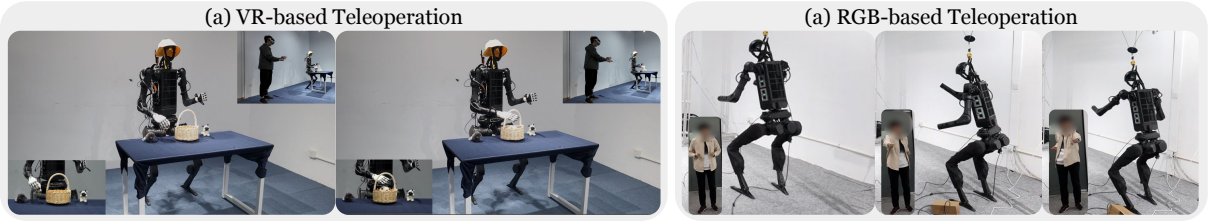


Figure 9.10. More physical teleoperation showcases.

The detailed performance of 4 tasks is documented in Table 9.17

Metrics	Catch-Release			Squat			Hammer-Catch			Rock-Paper-Scissors		
(a) Ablation on Data size												
	25%data	50%data	100%data	25%data	50%data	100%data	25%data	50%data	100%data	25%data	50%data	100%data
MSE Loss	3.01E-3	3.04E-4	9.89E-5	1.25E-4	1.10E-4	7.07E-5	2.18E-2	1.56E-2	3.29E-4	2.72E-2	1.39E-2	1.60E-3
Succ rate	1/10	3/10	6/10	9/10	10/10	10/10	3/10	6/10	6/10	3/10	9/10	10/10
(b) Ablation on Sequence observation/action												
	Si-O-Si-A	Se-O-Se-A	Si-O-Se-A	Si-O-Si-A	Se-O-Se-A	Si-O-Se-A	Si-O-Si-A	Se-O-Se-A	Si-O-Se-A	Si-O-Si-A	Se-O-Se-A	Si-O-Se-A
MSE Loss	2.52E-4	1.47E-4	9.89E-5	5.18E-5	9.60E-5	7.07E-5	2.22E-4	3.62E-4	3.29E-4	1.43E-3	3.36E-3	1.60E-3
Succ rate	3/10	7/10	6/10	10/10	10/10	10/10	5/10	9/10	6/10	10/10	9/10	10/10
(c) Ablation on BC/DDIM/DDPM												
	BC	DP-DDIM	DP-DDPM	BC	DP-DDIM	DP-DDPM	BC	DP-DDIM	DP-DDPM	BC	DP-DDIM	DP-DDPM
MSE Loss	1.39E-3	4.79E-5	9.89E-5	6.24E-4	6.42E-5	7.07E-5	4.50E-3	3.41E-4	3.29E-4	1.46E-2	2.42E-3	1.60E-3
Succ rate	0/10	6/10	6/10	3/10	10/10	10/10	0/10	5/10	6/10	1/10	10/10	10/10

Table 9.17. Quantitative LfD autonomous agents performance for 4 tasks.

9.5.11 Sim2real Training Hyperparameters

The hyperparameters for our RL/DAgger policy training are detailed in Table 9.18 below.

Hyperparameters	**Values**
Batch size	64
Discount factor (γ)	0.99
Learning rate	0.001
Clip param	0.2
Entropy coef	0.005
Max grad norm	0.2
Value loss coef	1
Entropy coef	0.005
Init noise std (RL)	1.0
Init noise std (DAgger)	0.001
Num learning epochs	5
MLP size	[512, 256, 128]

Table 9.18. Hyperparameters

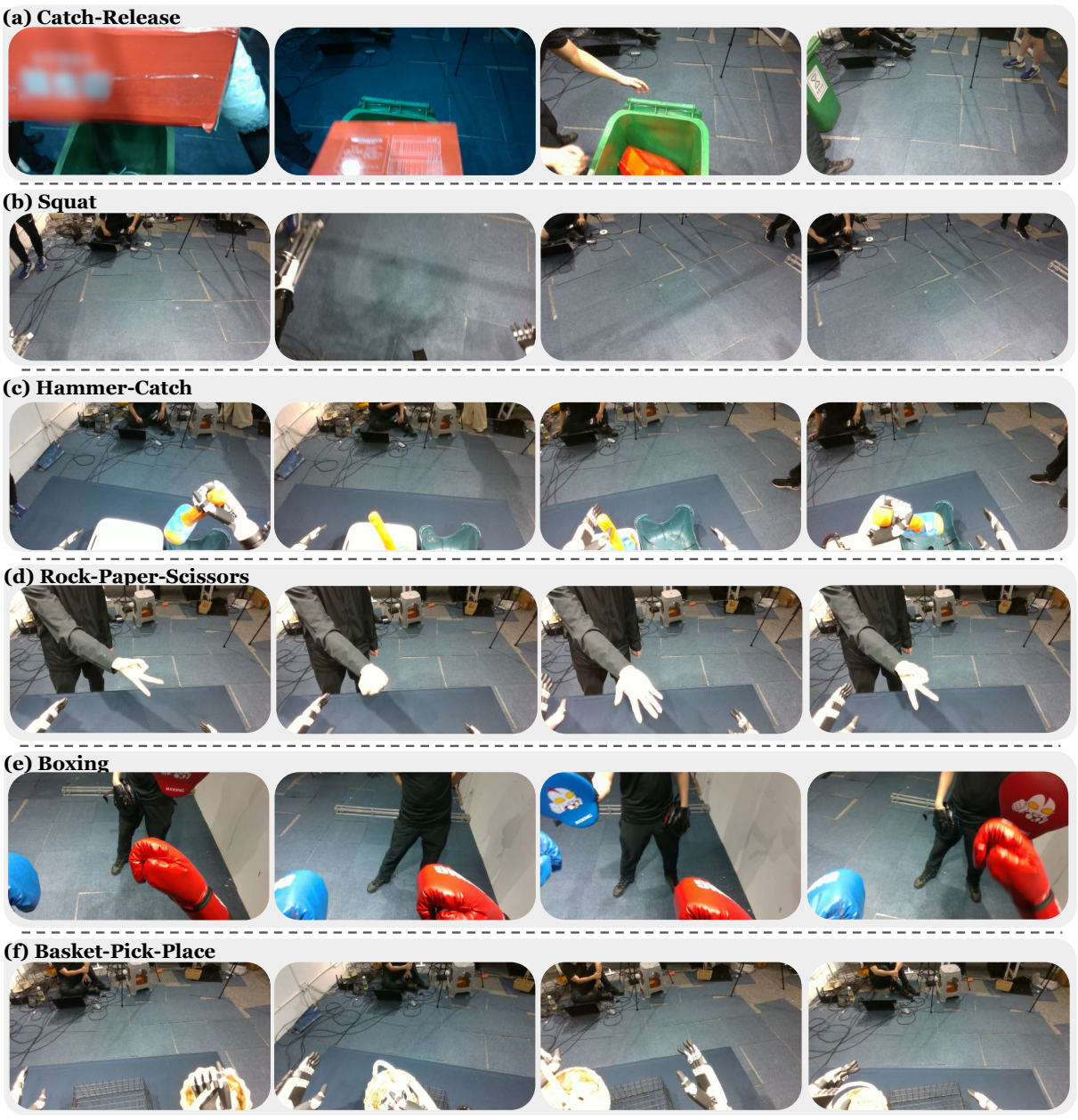


Figure 9.11. *OmniH2O-6* dataset.

9.5.12 LfD Hyperparameters

In order to make the robot autonomous, we have developed a Learning from Demonstration (LfD) approach utilizing a diffusion policy that learns from a dataset we collected. The default training hyperparameters are shown below in Table 9.19.

9.5.13 GPT-4o Prompt Example

Here is the example prompt we use for **Autonomous Boxing** task:

Hyperparameter	Default Value
Batch Size	32
Observation Horizon	1
Action Horizon	8
Prediction Horizon	16
Policy Dropout Rate	0.0
Dropout Rate (State Encoder)	0.0
Image Dropout Rate	0.0
Weight Decay	1E-5
Image Output Size	32
State Noise	0.0
Image Gaussian Noise	0.0
Image Masking Probability	0.0
Image Patch Size	16
Number of Diffusion Iterations	100

Table 9.19. Training Hyperparameters for the Lfd Training

You're a humanoid robot equipped with a camera slightly tilted downward on your head, providing a first-person perspective. I am assigning you a task: when a blue target appears in front of you, extend and then retract your left fist. When a red target appears, do the same with your right fist. If there is no target in front, remain stationary. I will provide you with three options each time: move your left hand forward, move your right hand forward, or stay motionless. You should directly respond with the corresponding options A, B, or C based on the current image. Note that, yourself is also wearing blue left boxing glove and right red boxing glove, please do not recognize them as the boxing target. Now, based on the current image, please provide me with the A, B, C answers.

For **Autonomous Greetings with Human** Task, our prompt is:

You are a humanoid robot equipped with a camera slightly tilted downward on your head, providing a first-person perspective. I am assigning you a new task to respond to human gestures in front of you. Remember, the person is standing facing you, so be mindful of their gestures. If the person extends their right hand to shake hands with you, use your right hand to shake their right hand (Option A). If the person opens both arms wide for a hug, open your arms wide to reciprocate the hug (Option B). If you see the person waving his hand as a gesture to say goodbye, respond by waving back (Option C). If no significant gestures are made, remain stationary (Option D). Respond directly with the corresponding options A, B, C, or D based on the current image and observed gestures. Directly reply with A, B, C, or D only, without any additional characters.

It is worth mentioning that we can use GPT-4 not only to choose motion primitive but also to directly generate the motion goal. The following prompt exemplifies this process:

You are a humanoid robot equipped with a camera slightly tilted downward on your head, providing a first-person perspective. I am assigning you a new task to respond to human gestures in front of you. If the person extends his left hand for a handshake, extend your left hand to reciprocate. If they extend their right hand, respond by extending your right hand. If the person opens both arms wide for a hug, open your arms wide to reciprocate the hug. If no significant gestures are made, remain stationary. Respond 6 numbers to represent the desired left and right hand 3D position with respect to your root position. For example: [0.25, 0.2, 0.3, 0.15, -0.19, 0.27] means the desired position of the left hand is 0.25m forward, 0.2m left, and 0.3m high compared to pelvis position, and the desired position of the right hand is 0.15m forward, 0.19m right and 0.27m high compared to pelvis position. The default stationary position should be (0.2, 0.2, 0.2, 0.2, -0.2, 0.2). Now please respond the 6d array based on the image to respond to the right hand shaking, left hand shaking, and hugging.

9.6 Limitations and Future Work

Summary. OmniH2O enables dexterous whole-body humanoid loco-manipulation via teleoperation, designs universal control interfaces, facilitates scalable demonstration collection, and empowers humanoid autonomy via frontier models or humanoid learning from demonstrations.

Limitations. One limitation of our system is the requirement of robot root odometry to transfer pose estimation from teleoperation interfaces to motion goals in the robot frame. Results from VIO can be noisy or even discontinuous, causing the motion goals to deviate from desired control. Another limitation is safety; although the OmniH2O policy has shown great robustness, we do not have guarantees or safety checks for extreme disturbances or out-of-distribution motion goals (*e.g.*, large discontinuity in motion goals). Future work could also focus on the design of the teleoperation system to allow the humanoid to traverse stairs with only sparse upper-body motion goals. Another interesting direction is improving humanoid learning from demonstrations by incorporating more sensors (*e.g.*, LiDAR, wrist cameras, tactile sensors) and better learning algorithms. We hope that our work spurs further efforts toward robust and scalable humanoid teleoperation and learning.

Part V

Conclusion and Future Work

Chapter 10

Conclusions and Future Work

10.1 Why Humanoid?

Humanoids have long captured our imagination as the ultimate physical embodiment of general intelligence. Their morphology — walking upright, using hands, interacting with complex environments designed for humans — makes them uniquely positioned to tackle a wide array of tasks across unstructured, human-centered settings. Unlike wheeled robots or single-purpose manipulators, humanoids possess the physical degrees of freedom necessary to operate tools, navigate diverse terrains, and perform tasks that require coordinated whole-body movement.

However, humanoids are incredibly complex, both from a software and hardware perspective. Compared to mobile manipulators that has wheels and two arms and other form factors, what is Humanoid's true strength? First, humanoids can walk up stairs, open doors, climb ladders, and reach high shelves — tasks that are difficult or impossible for wheeled or quadrupedal platforms. The more important factor is data: human motion, interaction, and behavior have been studied and recorded extensively across decades of research in computer vision, animation, and biomechanics. In this thesis, we can see that simulated humanoids benefit from large-scale human motion data to create interesting behaviors and animations. Real-world humanoids, similarly, benefit tremendously from existing human motion data. Crucially, the existing human datasets are directly applicable to humanoids due to the shared morphology. This opens up the possibility of leveraging vast amounts of human-centric data to bootstrap control, learning, and perception models. In contrast, form factors that deviate from the human body often require collecting task- or platform-specific data from scratch, limiting scalability.

10.2 Part I and II: How to Build the Behavioral Foundational Model

Throughout this thesis, we proposed a systematic path toward constructing a behavioral foundational model for humanoids: a model that encapsulates broad motor skills and can be reused for dexterous and perceptive tasks. This is akin to baby animals being pre-loaded with motor and behavioral priors at birth. The journey begins with motion imitation. By training reinforcement learning (RL) policies to imitate large-scale kinematic motion datasets, we enable humanoids to acquire a wide repertoire of dynamic behaviors — walking, jumping, balancing, reaching, grasping — taking the first step towards moving like humans. I feel like kinematic motion tracking, while not the final task to solve, is a great proxy task to verify that we are obtaining the necessary motor skills to perform diverse human actions.

We then distilled these capabilities into a compact, physics-based latent space of motor skills. This latent representation acts as a prior, defining a structured, human-like action space for downstream learn-

ing, enabling policies to explore like humans. Random exploration in this latent space leads to natural behaviors, dramatically improving the efficiency and realism of training new controllers.

10.3 Part III: Leveraging Behavioral Prior for Dexterous and Perceptive Tasks

Building on this control prior, we extended humanoid capabilities to dexterous object manipulation and vision-based control, learning to grasp like humans. By leveraging the motor latent space, we solved challenging grasping and manipulation tasks without requiring paired demonstrations. By integrating active perception, we enabled humanoids to search, recognize, and interact with objects in cluttered, realistic scenes.

10.4 Part IV: Transferring to the Real Humanoid

Finally, we demonstrated that these behaviors can transfer to real-world humanoids through carefully designed retargeting and sim-to-real strategies. We transfer PHC's pipeline to the real humanoid first via retargeting and a "sim-to-data" pipeline to find motion that can be performed by the robot. Then, we train our controllers with sim-to-real modifications such as real-world compatible state-space design, teacher-student distillation, and reward engineering. Each one of these steps can be arduous and involves days to weeks of tuning and testing. We verify that the pipeline designed for animation and vision (PHC) can be largely transferred to real humanoid robots. While we are not at feature parity with the simulation-only policies, we have taken first steps toward that goal.

10.5 Future Directions

While this thesis represents a step toward universal humanoid control, it also reveals new frontiers and challenges that must be addressed.

10.5.1 Scaling to Richer Behavioral Repertoires

Our current motion priors are primarily trained on motion datasets that focus on full-body movements. Extending to richer interactions — tool use, terrain traversal — will require scaling to even broader datasets, including multi-agent interactions and and fine-grained manipulation. Capturing and learning from human interaction with dynamic environments remains an open challenge. Our motor latent space is also not interpretable, making it hard to to be used to zero-shot. The two-stage learning process is also quite cumbersome, and a one-stage process would be preferred. I feel like such a motor latent space will be the key interface between System I (fast, reactive, intuitive motor control) and System II (deliberative, symbolic planning). Enabling communication and coordination across these systems, through a structured, compositional motor space, could be a key step toward building truly general-purpose embodied intelligence.

10.5.2 Closing the Loop Between Perception and Planning

Although we demonstrated perception-guided control in dexterous tasks, future humanoids must close the loop between perception, prediction, and planning at every level. This entails learning not only reactive policies, but also predictive models of the environment, task affordances, and object dynamics

— enabling anticipatory and strategic behavior. Integrating model-based planning with the behavioral prior could unlock new levels of autonomy and adaptability.

10.5.3 Sim-to-Real Transfer

At the moment, sim-to-real transfer is more art than science. Sim-to-real transfer requires more 10 to 20 reward terms, and each terms' coefficient requires careful tuning. How to automatically tune these rewards for better sim-to-real transfer is an open question. Also, as simulators become faster, more diverse, and differentiable, how to alleviate the dependency on a single simulator (*e.g*. IsaacGym) and learn across simulators is an open question.

10.5.4 Robustness and Safety in Open-World Settings

Real-world environments are unpredictable, cluttered, and filled with unseen situations. Future work must focus on making humanoid controllers robust to perturbations, sensor noise, model mismatch, and hardware failures. Developing scalable methods for online adaptation, uncertainty estimation, and safe exploration will be crucial for reliable deployment outside controlled lab settings.

10.5.5 Self-Supervised and Continual Learning

Finally, to truly achieve universal control, humanoids must not only perform tasks but also improve over time. Future systems should leverage self-supervised signals to refine their models, continually adapt to new tasks and environments, and retain previously learned behaviors without catastrophic forgetting. While sim-to-real has achieved impressive capabilities when deployed zero-shot, learning in real-world will be necessary for robots to generalize and adapt to new tasks.

While this thesis lays the foundation for scalable humanoid control, but the journey toward creating general-purpose, intelligent humanoids has just begun.

Bibliography

[1] Apple vision pro. https://www.apple.com/apple-vision-pro/. Accessed: 2023-10-19.

[2] Introducing project aria, from meta. https://www.projectaria.com/. Accessed: 2023-10-17.

[3] Meta quest 2: Immersive all-in-one VR headset. https://www.meta.com/ie/quest/products/quest-2/. Accessed: 2023-11-16.

[4] Meta quest 3: New mixed reality VR headset – shop now. https://www.meta.com/ie/quest/quest-3/. Accessed: 2023-10-17.

[5] Dexterous hand series. https://www.shadowrobot.com/dexterous-hand-series/, 19 sep 2023. Accessed: 2024-5-13.

[6] D. Abel, Y. Jinnai, S. Y. Guo, G. Konidaris, and M. Littman. Policy and value transfer in lifelong reinforcement learning. In *International conference on machine learning*, pages 20–29. PMLR, 2018.

[7] H. Akada, J. Wang, S. Shimada, M. Takahashi, C. Theobalt, and V. Golyanik. Unrealego: A new dataset for robust egocentric 3d human motion capture. *arXiv preprint arXiv:2208.01633*, 2022.

[8] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[9] M. Al Borno, M. De Lasa, and A. Hertzmann. Trajectory optimization for full-body movements with complex contacts. *IEEE transactions on visualization and computer graphics*, 19(8):1405–1414, 2012.

[10] S. Aliakbarian, P. Cameron, F. Bogo, A. Fitzgibbon, and T. J. Cashman. Flag: Flow-based 3d avatar generation from sparse observations. *arXiv preprint arXiv:2203.05789*, 2022.

[11] S. Aliakbarian, F. Saleh, D. Collier, P. Cameron, and D. Cosker. Hmd-nemo: Online 3d avatar motion generation from sparse observations. *arXiv preprint arXiv:2308.11261*, 2023.

[12] A. Arnab, C. Doersch, and A. Zisserman. Exploiting temporal context for 3d human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3395–3404, 2019.

[13] K. J. Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.

[14] K. Ayusawa and E. Yoshida. Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization. *IEEE Transactions on Robotics*, 33(6):1343–1357, 2017.

[15] J. Bae, J. Won, D. Lim, C.-H. Min, and Y. M. Kim. Pmp: Learning to physically interact with environments using part-wise motion priors. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023.

[16] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter. Alma-articulated locomotion and manipulation for a torque-controllable robot. In *2019 International conference on robotics and automation (ICRA)*, pages 8477–8483. IEEE, 2019.

[17] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48. Association for Computing Machinery, 2009.

[18] K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019.

[19] K. Bergamin, S. Clavet, D. Holden, and J. Richard Forbes. Drecon: Data-driven responsive control of physics-based characters. *ACM Trans. Graph.*, 38:11, 2019.

[20] G. Berseth, C. Xie, P. Cernek, and M. Van de Panne. Progressive reinforcement learning with distillation for multi-skilled motion control. *arXiv preprint arXiv:1802.04765*, 2018.

[21] Beta Program. Unity real-time development platform. https://unity.com/. Accessed: 2023-11-18.

[22] M. J. Black, P. Patel, J. Tesch, and J. Yang. Bedlam: A synthetic dataset of bodies exhibiting detailed lifelike animated motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8726–8737, 2023.

[23] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image. *Lect. Notes Comput. Sci.*, 9909 LNCS:561–578, 2016.

[24] S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, M. Wulfmeier, M. Neunert, B. Moran, N. Siegel, A. Huber, F. Romano, N. Batchelor, F. Casarini, J. Merel, R. Hadsell, and N. Heess. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors. *arXiv preprint arXiv:2203.17138*, 2022.

[25] S. Brahmbhatt, C. Ham, C. C. Kemp, and J. Hays. ContactDB: Analyzing and predicting grasp contact via thermal imaging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] J. Braun, S. Christen, M. Kocabas, E. Aksan, and O. Hilliges. Physically plausible full-body hand-object interaction synthesis. *I3DV*, 2024.

[27] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[28] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[29] M. A. Brubaker, L. Sigal, and D. J. Fleet. Estimating contact dynamics. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2389–2396. Ieee, 2009.

[30] Bullet Physics. humanoid urdf in bullet3, 2023. Accessed: 2024-03-01.

[31] V. Caggiano, S. Dasari, and V. Kumar. Myodex: Generalizable representations for dexterous physiological manipulation. 2022.

[32] L. Campanaro, S. Gangapurwala, W. Merkt, and I. Havoutis. Learning and deploying robust locomotion policies with minimal dynamics randomization, 2023.

[33] J. Cao, H. Tang, H.-S. Fang, X. Shen, C. Lu, and Y.-W. Tai. Cross-domain adaptation for animal pose estimation. *arXiv preprint arXiv:1908.05806*, 2019.

[34] J. Cao, X. Weng, R. Khirodkar, J. Pang, and K. Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. *arXiv preprint arXiv:2203.14360*, 2022.

[35] Z. Cao, I. Radosavovic, A. Kanazawa, and J. Malik. Reconstructing hand-object interactions in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12417–12426, 2021.

[36] J. Chagas Vaz, D. Wallace, and P. Y. Oh. Humanoid loco-manipulation of pushed carts utilizing virtual reality teleoperation. In *ASME International Mechanical Engineering Congress and Exposition*, volume 85628, page V07BT07A027. American Society of Mechanical Engineers, 2021.

[37] J. Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. Xu. Language to action: Towards interactive task learning with physical agents. In *IJCAI*, pages 2–9, 2018.

[38] Y.-W. Chao, J. Yang, W. Chen, and J. Deng. Learning to sit: Synthesizing human-chair interactions via hierarchical control. *ArXiv*, abs/1908.07423, 2019.

[39] C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, and J. Huang. Progressive feature alignment for unsupervised domain adaptation. *arXiv preprint arXiv:1811.08585*, 2018.

[40] T. Chen, M. Tippur, S. Wu, V. Kumar, E. Adelson, and P. Agrawal. Visual dexterity: In-hand reorientation of novel and complex object shapes. *Science Robotics*, 8(84):eadc9244, 2023.

[41] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation. *Conference on Robot Learning*, 2021.

[42] Z. Chen, S. Chen, E. Arlaud, I. Laptev, and C. Schmid. Vividex: Learning vision-based dexterous manipulation from human videos. *arXiv preprint arXiv:2404.15709*, 2024.

[43] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024.

[44] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: open-source tele-operation with vision, 2024.

[45] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[46] N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke. Physics-based motion capture imitation with deep reinforcement learning. *Proceedings - MIG 2018: ACM SIGGRAPH Conference on Motion, Interaction, and Games*, 2018.

[47] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[48] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[49] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[50] S. Christen, L. Feng, W. Yang, Y.-W. Chao, O. Hilliges, and J. Song. Synh2r: Synthesizing hand-object motions for learning human-to-robot handovers. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3168–3175. IEEE, 2024.

[51] S. Christen, M. Kocabas, E. Aksan, J. Hwangbo, J. Song, and O. Hilliges. D-grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20545–20554, 2021.

[52] R. Cisneros-Limón, A. Dallard, M. Benallegue, K. Kaneko, H. Kaminaga, P. Gergondet, A. Tanguy, R. P. Singh, L. Sun, Y. Chen, et al. A cybernetic avatar system to embody human telepresence for connectivity, exploration, and skill transfer. *International Journal of Social Robotics*, pages 1–28, 2024.

[53] CMU. Cmu graphics lab motion capture database, 2002.

[54] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016.

[55] S. Dafarra, U. Pattacini, G. Romualdi, L. Rapetti, R. Grieco, K. Darvish, G. Milani, E. Valli, I. Sorrentino, P. M. Viceconte, et al. icub3 avatar system: Enabling remote fully immersive embodiment of humanoid robots. *Science Robotics*, 9(86):eadh3834, 2024.

[56] D. Dajles, F. Siles, et al. Teleoperation of a humanoid robot using an optical motion capture system. In *2018 IEEE International Work Conference on Bioinspired Intelligence (IWOBI)*, pages 1–8. IEEE, 2018.

[57] Damiao. Static, dynamic, fictitious, strong, 2018.

[58] J. Dao, K. Green, H. Duan, A. Fern, and J. Hurst. Sim-to-real learning for bipedal locomotion under unsensed dynamic loads. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10449–10455. IEEE, 2022.

[59] K. Darvish, L. Penco, J. Ramos, R. Cisneros, J. Pratt, E. Yoshida, S. Ivaldi, and D. Pucci. Teleoperation of humanoid robots: A survey. *IEEE Transactions on Robotics*, 2023.

[60] K. Darvish, Y. Tirupachuri, G. Romualdi, L. Rapetti, D. Ferigo, F. J. A. Chavez, and D. Pucci. Whole-body geometric retargeting for humanoid robots. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 679–686. IEEE, 2019.

[61] S. Dasari, A. Gupta, and V. Kumar. Learning dexterous manipulation from exemplar object trajectories and pre-grasps. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3889–3896. IEEE, 2023.

[62] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44:3366–3385, 2022.

[63] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[64] A. Di Fava, K. Bouyarmane, K. Chappellet, E. Ruffaldi, and A. Kheddar. Multi-contact motion retargeting from human to humanoid robot. In *2016 IEEE-RAS 16th international conference on humanoid robots (humanoids)*, pages 1081–1086. IEEE, 2016.

[65] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[66] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[67] Y. Du, R. Kips, A. Pumarola, S. Starke, A. Thabet, and A. Sanakoyeu. Avatars grow legs: Generating smooth human motion from sparse tracking inputs with diffusion model. *arXiv preprint arXiv:2304.08577*, 2023.

[68] H. Duan, B. Pandit, M. S. Gadde, B. J. van Marum, J. Dao, C. Kim, and A. Fern. Learning vision-based bipedal locomotion for challenging terrain. *arXiv preprint arXiv:2309.14594*, 2023.

[69] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *arXiv preprint arXiv:1702.03118*, 2017.

[70] M. Elobaid, Y. Hu, G. Romualdi, S. Dafarra, J. Babic, and D. Pucci. Telexistence and teleoperation for walking humanoid robots. In *Intelligent Systems and Applications: Proceedings of the 2019 Intelligent Systems Conference (IntelliSys) Volume 2*, pages 1106–1121. Springer, 2020.

[71] C. Eppner, A. Murali, C. Garrett, R. O'Flaherty, T. Hermans, W. Yang, and D. Fox. scene_synthesizer: A python library for procedural scene generation in robot manipulation. *Journal of Open Source Software*, 2024.

[72] Z. Fan, M. Parelli, M. E. Kadoglou, M. Kocabas, X. Chen, M. J. Black, and O. Hilliges. Hold: Category-agnostic 3d reconstruction of interacting hands and objects from video. *arXiv preprint arXiv:2311.18448*, 2023.

[73] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 2023.

[74] R. M. French and N. Chater. Using noise to compute error surfaces in connectionist networks: a novel means of reducing catastrophic forgetting. *Neural Comput.*, 14:1755–1769, 2002.

[75] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra. First-person tele-operation of a humanoid robot. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 997–1002. IEEE, 2015.

[76] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: Learning a unified policy for manipulation and locomotion. *arXiv preprint arXiv:2210.10044*, 2022.

[77] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[78] K. Fukushima. Cognitron: a self-organizing multilayered neural network. *Biol. Cybern.*, 20:121–136, 1975.

[79] L. Fussell, K. Bergamin, and D. Holden. Supertrack: motion tracking for physically simulated characters using supervised learning. *ACM Trans. Graph.*, 40:1–13, 2021.

[80] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 409–419, 2018.

[81] Z. Geng, K. Sun, B. Xiao, Z. Zhang, and J. Wang. Bottom-up human pose estimation via disentangled keypoint regression. *arXiv preprint arXiv:2104.02300*, 2021.

[82] G. V. Georgakis, R. Li, S. Karanam, T. Chen, J. Kosecka, and Z. Wu. Hierarchical kinematic human mesh recovery. *ArXiv*, abs/2003.04232, 2020.

[83] A. Ghosh, R. Dabral, V. Golyanik, C. Theobalt, and P. Slusallek. Imos: Intent-driven full-body motion synthesis for human-object interactions. In *Eurographics*, 2023.

[84] K. Gong, B. Li, J. Zhang, T. Wang, J. Huang, M. B. Mi, J. Feng, and X. Wang. Posetriplet: Co-evolving 3d human pose estimation, imitation, and hallucination under self-supervision. *CVPR*, 2022.

[85] J. W. Grizzle, J. Hurst, B. Morris, H.-W. Park, and K. Sreenath. Mabel, a new robotic bipedal walker and runner. In *2009 American Control Conference*, pages 2030–2036. IEEE, 2009.

[86] R. A. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[87] C. Guo, X. Zuo, S. Wang, and L. Cheng. Tm2t: Stochastic and tokenized modeling for the reciprocal generation of 3d human motions and texts. *arXiv preprint arXiv:2207.01696*, 2022.

[88] E. Gärtner, M. Andriluka, H. Xu, and C. Sminchisescu. Trajectory optimization for physics-based reconstruction of 3d human pose from monocular video. *arXiv preprint arXiv:2205.12292*, 2022.

[89] T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018.

[90] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *arXiv preprint arXiv:2304.13653*, 2023.

[91] I. Habibie, W. Xu, D. Mehta, G. Pons-Moll, and C. Theobalt. In the wild human pose estimation using explicit 2d features and intermediate 3d representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10897–10906, Jun. 2019.

[92] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox. Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9164–9170. IEEE, 2020.

[93] L. Hasenclever, F. Pardo, R. Hadsell, N. Heess, and J. Merel. CoMic: Complementary task learning & mimicry for reusable skills. In H. D. Iii and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4105–4115. PMLR, 2020.

[94] M. Hassan, Y. Guo, T. Wang, M. Black, S. Fidler, and X. B. Peng. Synthesizing physical character-scene interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–9, 2023.

[95] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[96] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, et al. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. *arXiv preprint arXiv:2502.01143*, 2025.

[97] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. In *arXiv*, 2024.

[98] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024.

[99] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, C. Liu, G. Shi, X. Wang, L. Fan, and Y. Zhu. Hover: Versatile neural whole-body controller for humanoid robots. *arXiv preprint arXiv:2410.21229*, 2024.

[100] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi. Agile but safe: Learning collision-free high-speed legged locomotion, 2024.

[101] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[102] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenaka. The development of honda humanoid robot. In *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*, volume 2, pages 1321–1326. IEEE, 1998.

[103] M. Hirschmanner, C. Tsiourti, T. Patten, and M. Vincze. Virtual reality teleoperation of a humanoid robot using markerless human upper body pose imitation. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 259–265. IEEE, 2019.

[104] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[105] D. D. Hoffman, M. Singh, and C. Prakash. The interface theory of perception. *Psychonomic bulletin & review*, 22:1480–1506, 2015.

[106] T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa. Predictive Sampling: Real-time Behaviour Synthesis with MuJoCo. dec 2022.

[107] K. Hu, C. Ott, and D. Lee. Online human walking imitation in task and joint space based on quadratic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3458–3464. IEEE, 2014.

[108] B. Huang, L. Pan, Y. Yang, J. Ju, and Y. Wang. Neural mocon: Neural motion control for physically plausible human motion capture. *arXiv preprint arXiv:2203.14065*, 2022.

[109] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. *arXiv preprint arXiv:2409.01652*, 2024.

[110] A. Inc. Estimating camera pose with arkit. https://developer.apple.com/documentation/arkit/arcamera, 2021. Accessed: 2021-03-16.

[111] A. Inc. Scanning and detecting 3d objects with arkit. https://developer.apple.com/documentation/arkit/content_anchors/scanning_and_detecting_3d_objects, 2021. Accessed: 2021-03-16.

[112] Inspire-robots. Smaller and higher-precision motion control experts, 2018.

[113] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[114] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36:1325–1339, 2014.

[115] Y. Ishiguro, K. Kojima, F. Sugai, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. High speed whole body dynamic motion experiment with real time master-slave humanoid robot system. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5835–5841. IEEE, 2018.

[116] Y. Ishiguro, T. Makabe, Y. Nagamatsu, Y. Kojio, K. Kojima, F. Sugai, Y. Kakiuchi, K. Okada, and M. Inaba. Bilateral humanoid teleoperation system using whole-body exoskeleton cockpit tablis. *IEEE Robotics and Automation Letters*, 5(4):6419–6426, 2020.

[117] M. Isogawa, Y. Yuan, M. O'Toole, and K. M. Kitani. Optical non-line-of-sight physics-based 3d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7013–7022, 2020.

[118] Z. Jia, X. Li, Z. Ling, S. Liu, Y. Wu, and H. Su. Improving policy optimization with generalist-specialist learning. *arXiv preprint arXiv:2206.12984*, 2022.

[119] B. Jiang, X. Chen, W. Liu, J. Yu, G. Yu, and T. Chen. Motiongpt: Human motion as a foreign language. *arXiv preprint arXiv:2306.14795*, 2023.

[120] H. Jiang, S. Liu, J. Wang, and X. Wang. Hand-object contact consistency reasoning for human grasps generation. In *ICCV*, 2021.

[121] H. Jiang and K. Grauman. Seeing invisible poses: Estimating 3d body pose from egocentric video. *arXiv preprint arXiv:1603.07763*, pages 3501–3509, 2016.

[122] J. Jiang, P. Streli, H. Qiu, A. Fender, L. Laich, P. Snape, and C. Holz. Avatarposer: Articulated full-body pose tracking from sparse motion sensing. *arXiv preprint arXiv:2207.13784*, 2022.

[123] Y. Jiang, M. Guo, J. Li, I. Exarchos, J. Wu, and C. K. Liu. Dash: Modularized human manipulation simulation with vision and language for embodied ai. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 1–12, 2021.

[124] Jocher, Glenn and Chaurasia, Ayush and Qiu, Jing. Yolov8.

[125] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018.

[126] A. Kanazawa, J. Zhang, P. Felsen, and J. Malik. Learning 3d human dynamics from video. *CoRR*, abs/1812.01601, 2018.

[127] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[128] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[129] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, pages 1–14, 2014.

[130] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[131] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. U. S. A.*, 114(13):3521–3526, 2017.

[132] M. Kocabas, N. Athanasiou, and M. J. Black. Vibe: Video inference for human body pose and shape estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 5252–5262, 2020.

[133] G. Koch, R. Zemel, R. Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, pages 1–30. Lille, 2015.

[134] J. Koenemann, F. Burget, and M. Bennewitz. Real-time imitation of human whole-body motions by humanoids. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2812. IEEE, 2014.

[135] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2252–2261, 2019.

[136] D. Kulić, G. Venture, K. Yamane, E. Demircan, I. Mizuuchi, and K. Mombaur. Anthropomorphic movement analysis and synthesis: A survey of methods and applications. *IEEE Transactions on Robotics*, 32(4):776–795, 2016.

[137] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[138] S. Lee, S. Starke, Y. Ye, J. Won, and A. Winkler. Questenvsim: Environment-aware simulated motion tracking from sparse sensors. *arXiv preprint arXiv:2306.05666*, 2023.

[139] C. Lenz, M. Schwarz, A. Rochow, B. Pätzold, R. Memmesheimer, M. Schreiber, and S. Behnke. Nimbro wins ana avatar xprize immersive telepresence competition: Human-centric evaluation and lessons learned. *International Journal of Social Robotics*, pages 1–25, 2023.

[140] J. Li, C. K. Liu, and J. Wu. Ego-body pose estimation via ego-head pose estimation. *arXiv preprint arXiv:2212.04636*, 2022.

[141] J. Li, R. Villegas, D. Ceylan, J. Yang, Z. Kuang, H. Li, and Y. Zhao. Task-generic hierarchical human motion prior using vaes. *arXiv preprint arXiv:2106.04004*, 2021.

[142] J. Li, J. Wu, and C. K. Liu. Object motion guided human motion synthesis. *ACM Transactions on Graphics (TOG)*, 42(6):1–11, 2023.

[143] J. Li, C. Xu, Z. Chen, S. Bian, L. Yang, and C. Lu. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. *arXiv preprint arXiv:2011.14672*, 2020.

[144] Q. Li, J. Wang, C. C. Loy, and B. Dai. Task-oriented human-object interactions generation with implicit neural representations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3035–3044, 2024.

[145] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *ACCV*, 2014.

[146] Z. Li, J. Liu, Z. Zhang, S. Xu, and Y. Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. *arXiv preprint arXiv:2208.00571*, 2022.

[147] Z. Li, X. Cheng, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. 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.

[148] Z. Li, X. B. Peng, P. Abbeel, S. Levine, G. Berseth, and K. Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *arXiv preprint arXiv:2401.16889*, 2024.

[149] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids, 2025.

[150] T. Lin, Z.-H. Yin, H. Qi, P. Abbeel, and J. Malik. Twisting lids off with two hands. *arXiv:2403.02338*, 2024.

[151] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik. Learning visuotactile skills with two multifingered hands. *arXiv preprint arXiv:2404.16823*, 2024.

[152] H. Y. Ling, F. Zinno, G. Cheng, and M. Van De Panne. Character controllers using motion vaes. *ACM Trans. Graph.*, 39:12, 2020.

[153] H. Y. Ling, F. Zinno, G. H. Cheng, and M. V. D. Panne. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)*, 39:40:1–40:12, 2020.

[154] D. Liu, A. Lamb, K. Kawaguchi, A. Goyal, C. Sun, M. C. Mozer, and Y. Bengio. Discrete-valued neural communication. *arXiv preprint arXiv:2107.02367*, 2021.

[155] L. Liu and J. Hodgins. Learning basketball dribbling skills using trajectory optimization and deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[156] P. Liu, Y. Orru, C. Paxton, N. M. M. Shafiullah, and L. Pinto. Ok-robot: What really matters in integrating open-knowledge models for robotics. *arXiv preprint arXiv:2401.12202*, 2024.

[157] S. Liu, S. Tripathi, S. Majumdar, and X. Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3282–3292, 2022.

[158] S. Liu, G. Lever, Z. Wang, J. Merel, S. M. Ali Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki, N. Y. Siegel, L. Hasenclever, L. Marris, S. Tunyasuvunakool, H. Francis Song, M. Wulfmeier, P. Muller, T. Haarnoja, B. D. Tracey, K. Tuyls, T. Graepel, and N. Heess. From motor control to team play in simulated humanoid football. *arXiv preprint arXiv:2105.12196*, 2021.

[159] X. Liu and L. Yi. Geneoh diffusion: Towards generalizable hand-object interaction denoising via denoising diffusion. In *ICLR*, 2024.

[160] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6):1–16, 2015.

[161] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *ACM Trans. Graph.*, 34, 2015.

[162] M. Loper, N. Mahmoody, and M. J. Blackz. Mosh: Motion and shape capture from sparse markers. *ACM Trans. Graph.*, 33, 2014.

[163] C. Lu, X. Cheng, J. Li, S. Yang, M. Ji, C. Yuan, G. Yang, S. Yi, and X. Wang. Mobile-television: Predictive motion priors for humanoid whole-body control. *arXiv preprint arXiv:2412.07773*, 2024.

[164] T. Lucas, F. Baradel, P. Weinzaepfel, and G. Rogez. Posegpt: Quantization-based 3d human motion generation and forecasting. 2022.

[165] T. G. W. Lum, M. Matak, V. Makoviychuk, A. Handa, A. Allshire, T. Hermans, N. D. Ratliff, and K. Van Wyk. Dextrah-g: Pixels-to-action dexterous arm-hand grasping with geometric fabrics. *arXiv preprint arXiv:2407.02274*, 2024.

[166] Y.-S. Luo, J. H. Soeseno, T. P.-C. Chen, and W.-C. Chen. Carl: Controllable agent with reinforcement learning for quadruped locomotion. *arXiv preprint arXiv:2005.03288*, 2020.

[167] Z. Luo, J. Cao, S. Christen, A. Winkler, K. M. Kitani, and W. Xu. Omnigrasp: Grasping diverse objects with simulated humanoids. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[168] Z. Luo, J. Cao, R. Khirodkar, A. Winkler, K. Kitani, and W. Xu. Real-time simulated avatar from head-mounted sensors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 571–581, 2024.

[169] Z. Luo, J. Cao, K. Kitani, W. 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.

[170] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu. Universal humanoid motion representations for physics-based control. *arXiv preprint arXiv:2310.04582*, 2023.

[171] Z. Luo, S. A. Golestaneh, and K. M. Kitani. 3d human motion estimation via motion compression and refinement. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, November 2020.

[172] Z. Luo, R. Hachiuma, Y. Yuan, and K. Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. *NeurIPS*, 34:25019–25032, 2021.

[173] Z. Luo, S. Iwase, Y. Yuan, and K. Kitani. Embodied scene-aware human pose estimation. *NeurIPS*, 2022.

[174] Z. Luo, J. Wang, K. Liu, H. Zhang, C. Tessler, J. Wang, Y. Yuan, J. Cao, Z. Lin, F. Wang, et al. Smplolympics: Sports environments for physically simulated humanoids. *arXiv preprint arXiv:2407.00187*, 2024.

[175] Z. Luo, Y. Yuan, and K. M. Kitani. From universal humanoid control to automatic physically valid character creation. *arXiv preprint arXiv:2206.09286*, 2022.

[176] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. Amass: Archive of motion capture as surface shapes. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:5441–5450, 2019.

[177] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and Gavriel State. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

[178] P. Mandikal and K. Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning*, pages 651–661. PMLR, 2022.

[179] M. McCloskey and N. J. Cohen. *Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem*, volume 24, pages 109–165. Academic Press, 1989.

[180] Y. Meng, Z. Bing, X. Yao, K. Chen, K. Huang, Y. Gao, F. Sun, and A. Knoll. Preserving and combining knowledge in robotic lifelong reinforcement learning. *Nature Machine Intelligence*, pages 1–14, 2025.

[181] J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Reusable neural skill embeddings for vision-guided whole body movement and object manipulation. *ArXiv*, abs/1911.06636, 2019.

[182] J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018.

[183] J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control, 2018.

[184] J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Catch and carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM Trans. Graph.*, 39, 2020.

[185] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[186] F.-J. Montecillo-Puente, M. Sreenivasa, and J.-P. Laumond. On real-time whole-body human to humanoid motion transfer. 2010.

[187] G. Moon, J. Y. Chang, and K. M. Lee. Camera distance-aware top-down approach for 3d multi-person pose estimation from a single rgb image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10133–10142, 2019.

[188] G. Moon and K. M. Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *Eccv*, 2020.

[189] G. Moon, S. Saito, W. Xu, R. Joshi, J. Buffalini, H. Bellan, N. Rosen, J. Richardson, M. Mallorie, P. Bree, T. Simon, B. Peng, S. Garg, K. McPhail, and T. Shiratori. A dataset of relighted 3D interacting hands. In *NeurIPS Track on Datasets and Benchmarks*, 2023.

[190] T. Nagarajan, C. Feichtenhofer, and K. Grauman. Grounded human-object interaction hotspots from video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8688–8697, 2019.

[191] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines.

[192] S. Narasimhaswamy, U. Bhattacharya, X. Chen, I. Dasgupta, S. Mitra, and M. Hoai. Handiffuser: Text-to-image generation with realistic hand appearances. *arXiv preprint arXiv:2403.01693*, 2024.

[193] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. *arXiv preprint arXiv:2402.07872*, 2024.

[194] S. Nath, C. Peridis, E. Ben-Iwhiwhu, X. Liu, S. Dora, C. Liu, S. Kolouri, and A. Soltoggio. Sharing lifelong reinforcement learning knowledge via modulating masks. In *Conference on Lifelong Learning Agents*, pages 936–960. PMLR, 2023.

[195] E. Ng, D. Xiang, H. Joo, and K. Grauman. You2me: Inferring body pose in egocentric video via first and second person interactions. *CoRR*, abs/1904.09882, 2019.

[196] A. Noë. *Action in perception*. MIT press, 2004.

[197] J. K. O'regan and A. Noë. A sensorimotor account of vision and visual consciousness. *Behavioral and brain sciences*, 24(5):939–973, 2001.

[198] K. Otani and K. Bouyarmane. Adaptive whole-body manipulation in human-to-humanoid multi-contact motion retargeting. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 446–453. IEEE, 2017.

[199] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[200] X. Pan, N. Charron, Y. Yang, S. Peters, T. Whelan, C. Kong, O. Parkhi, R. Newcombe, and C. Y. Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. *arXiv preprint arXiv:2306.06362*, 2023.

[201] H. Park, R. Yu, and J. Lee. Multi-segment foot modeling for human animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*, number Article 16 in MIG '18, pages 1–10. Association for Computing Machinery, 2018.

[202] S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Trans. Graph.*, 38:1–11, 2019.

[203] Y. Park and P. Agrawal. Using apple vision pro to train and control robots, 2024.

[204] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3d hands, face, and body from a single image. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:10967–10977, 2019.

[205] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7753–7762, Jun. 2019.

[206] L. Penco, B. Clément, V. Modugno, E. M. Hoffman, G. Nava, D. Pucci, N. G. Tsagarakis, J.-B. Mouret, and S. Ivaldi. Robust real-time whole-body motion retargeting from human to humanoid. In *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pages 425–432. IEEE, 2018.

[207] L. Penco, K. Momose, S. McCrory, D. Anderson, N. Kitchel, D. Calvert, and R. J. Griffin. Mixed reality teleoperation assistance for direct control of humanoids. *IEEE Robotics and Automation Letters*, 2024.

[208] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37:143:1–143:14, 2018.

[209] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. 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.

[210] X. B. Peng, G. Berseth, and M. Van de Panne. Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 35(4):81, 2016.

[211] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne. Deeploco: dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.*, 36:1–13, 2017.

[212] X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. In *Advances in Neural Information Processing Systems*, pages 3681–3692, 2019.

[213] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *arXiv preprint arXiv:2205.01906*, 2022.

[214] X. B. Peng, A. Kanazawa, J. Malik, P. Abbeel, and S. Levine. Sfv: Reinforcement learning of physical skills from videos. In *SIGGRAPH Asia 2018 Technical Papers*, volume 37, page 178, New York, NY, USA, November 2018. Acm, ACM New York, NY, USA.

[215] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Trans. Graph.*, pages 1–20, 2021.

[216] M. Petrovich, M. J. Black, and G. Varol. Action-conditioned 3d human motion synthesis with transformer vae. *arXiv preprint arXiv:2104.05670*, 2021.

[217] J. L. Ponton, H. Yun, C. Andujar, and N. Pelechano. Combining motion matching and orientation prediction to animate avatars for consumer-grade vr devices. In *Computer Graphics Forum*, volume 41, pages 107–118. Wiley Online Library, 2022.

[218] O. Porges, M. Connan, B. Henze, A. Gigli, C. Castellini, and M. A. Roa Garzon. A wearable, ultralight interface for bimanual teleoperation of a compliant, whole-body-controlled humanoid robot. In *2019 International Conference on Robotics and Automation, ICRA 2019*. IEEE, 2019.

[219] S. Prokudin, C. Lassner, and J. Romero. Efficient learning on point clouds with basis point sets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4332–4341, 2019.

[220] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. In *Conference on Robot Learning*, pages 2549–2564. PMLR, 2023.

[221] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pages 570–587. Springer, 2022.

[222] Y. Qin, W. Yang, B. Huang, K. Van Wyk, H. Su, X. Wang, Y.-W. Chao, and D. Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.

[223] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Real-world humanoid locomotion with reinforcement learning. *Science Robotics*, 9(89):eadi9579, 2024.

[224] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik. Humanoid locomotion as next token prediction. 2024.

[225] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[226] J. Ramos and S. Kim. Humanoid dynamic synchronization through whole-body bilateral feedback teleoperation. *IEEE Transactions on Robotics*, 34(4):953–965, 2018.

[227] J. Ramos and S. Kim. Dynamic locomotion synchronization of bipedal robot and human operator via bilateral feedback teleoperation. *Science Robotics*, 4(35):eaav4282, 2019.

[228] O. E. Ramos, N. Mansard, O. Stasse, C. Benazeth, S. Hak, and L. Saab. Dancing humanoid robots: Systematic use of osid to compute dynamically consistent movements following a motion capture pattern. *IEEE Robotics & Automation Magazine*, 22(4):16–26, 2015.

[229] D. Rao, F. Sadeghi, L. Hasenclever, M. Wulfmeier, M. Zambelli, G. Vezzani, D. Tirumala, Y. Aytar, J. Merel, N. Heess, and R. Hadsell. Learning transferable motor skills with hierarchical latent mixture policies. *arXiv preprint arXiv:2112.05062*, 2021.

[230] M. Rayat Imtiaz Hossain and J. J. Little. Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 68–84, 2018.

[231] D. Rempe, T. Birdal, A. Hertzmann, J. Yang, S. Sridhar, and L. J. Guibas. Humor: 3d human motion model for robust pose estimation. *arXiv preprint arXiv:2105.04668*, 2021.

[232] D. Rempe, L. J. Guibas, A. Hertzmann, B. Russell, R. Villegas, and J. Yang. Contact and human dynamics from monocular video. *Lect. Notes Comput. Sci.*, 12350 LNCS:71–87, 2020.

[233] D. Rempe, Z. Luo, X. B. Peng, Y. Yuan, K. Kitani, K. Kreis, S. Fidler, and O. Litany. Trace and pace: Controllable pedestrian animation via guided trajectory diffusion. *arXiv preprint arXiv:2304.01893*, 2023.

[234] J. Ren, C. Yu, S. Chen, X. Ma, L. Pan, and Z. Liu. Diffmimic: Efficient motion mimicking with differentiable physics. *arXiv preprint arXiv:2304.03274*, 2023.

[235] H. Rhodin, C. Richardt, D. Casas, E. Insafutdinov, M. Shafiei, H.-P. Seidel, B. Schiele, and C. Theobalt. Egocap: egocentric marker-less motion capture with two fisheye cameras. *ACM Transactions on Graphics (TOG)*, 35(6):162, November 2016.

[236] G. Rogez, P. Weinzaepfel, and C. Schmid. LCR-Net++: Multi-person 2D and 3D Pose Detection in Natural Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[237] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. *ArXiv*, abs/2201.02610, 2022.

[238] S. Ross and D. Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668, 2010.

[239] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *arXiv preprint arXiv:1011.0686*, 2010.

[240] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *arXiv preprint arXiv:2109.11978*, 2021.

[241] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[242] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[243] T. N. Sainath, O. Vinyals, A. W. Senior, and H. Sak. Convolutional, long short-term memory, fully connected deep neural networks. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4580–4584, 2015.

[244] S. Schmitt, J. J. Hudson, A. Zidek, S. Osindero, C. Doersch, W. M. Czarnecki, J. Z. Leibo, H. Kuttler, A. Zisserman, K. Simonyan, and S. M. A. Eslami. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

[245] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

[246] M. Seo, S. Han, K. Sim, S. H. Bang, C. Gonzalez, L. Sentis, and Y. Zhu. Deep imitation learning for humanoid loco-manipulation through human teleoperation. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2023.

[247] K. Shaw, A. Agarwal, and D. Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *arXiv preprint arXiv:2309.06440*, 2023.

[248] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[249] S. Shimada, V. Golyanik, W. Xu, P. Pérez, and C. Theobalt. Neural monocular 3d human motion capture. *ACM Transactions on Graphics*, 40(4), aug 2021.

[250] S. Shimada, V. Golyanik, W. Xu, P. Pérez, and C. Theobalt. Neural monocular 3d human motion capture with physical awareness. *arXiv preprint arXiv:2105.01057*, 2021.

[251] S. Shimada, V. Golyanik, W. Xu, and C. Theobalt. Physcap: Physically plausible monocular 3d motion capture in real time. *arXiv preprint arXiv:2008.08880*, 2020.

[252] J. Siekmann, Y. Godse, A. Fern, and J. Hurst. Sim-to-real learning of all common bipedal gaits via periodic reward composition. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7309–7315. IEEE, 2021.

[253] J. Siekmann, K. Green, J. Warila, A. Fern, and J. Hurst. Blind bipedal stair traversal via sim-to-real reinforcement learning. *arXiv preprint arXiv:2105.08328*, 2021.

[254] D. L. Silver, Q. Yang, and L. Li. Lifelong machine learning systems: Beyond learning algorithms. In *AAAI Spring Symposium: Lifelong Machine Learning*, volume 13, 2013.

[255] H. G. Singh, A. Loquercio, C. Sferrazza, J. Wu, H. Qi, P. Abbeel, and J. Malik. Hand-object interaction pretraining from videos. *arXiv preprint arXiv:2409.08273*, 2024.

[256] R. Singh, A. Allshire, A. Handa, N. Ratliff, and K. Van Wyk. Dextrah-rgb: Visuomotor policies to grasp anything with dexterous hands. *arXiv preprint arXiv:2412.01791*, 2024.

[257] K. K. Somasundaram, J. Dong, H. Tang, J. Straub, M. Yan, M. Goesele, J. J. Engel, R. D. Nardi, and R. A. Newcombe. Project aria: A new tool for egocentric multi-modal ai research. *ArXiv*, abs/2308.13561, 2023.

[258] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[259] C. Stanton, A. Bogdanovych, and E. Ratanasena. Teleoperation of a humanoid robot using full-body motion capture, example movements, and machine learning. In *Proc. Australasian Conference on Robotics and Automation*, volume 8, page 51, 2012.

[260] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, Jun. 2018.

[261] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[262] I. Sárándi, A. Hermans, and B. Leibe. Learning 3d human pose estimation from dozens of datasets using a geometry-aware autoencoder to bridge between skeleton formats. *arXiv preprint arXiv:2212.14474*, 2022.

[263] I. Sárándi, T. Linder, K. O. Arras, and B. Leibe. Metrabs: Metric-scale truncation-robust heatmaps for absolute 3d human pose estimation. pages 1–14, 2020.

[264] S. Tachi, Y. Inoue, and F. Kato. Telesar vi: Telexistence surrogate anthropomorphic robot vi. *International Journal of Humanoid Robotics*, 17(05):2050019, 2020.

[265] O. Taheri, V. Choutas, M. J. Black, and D. Tzionas. GOAL: Generating 4D whole-body motion for hand-object grasping. In *CVPR*, 2022.

[266] O. Taheri, N. Ghorbani, M. J. Black, and D. Tzionas. Grab: A dataset of whole-body human grasping of objects. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 581–600. Springer, 2020.

[267] O. Taheri, Y. Zhou, D. Tzionas, Y. Zhou, D. Ceylan, S. Pirk, and M. J. Black. GRIP: Generating interaction poses using latent consistency and spatial cues. In *I3DV*, 2024.

[268] J. Tan, K. Liu, and G. Turk. Stable proportional-derivative controllers. *IEEE Computer Graphics and Applications*, 31(4):34–44, Jul. 2011.

[269] A. Tang, T. Hiraoka, N. Hiraoka, F. Shi, K. Kawaharazuka, K. Kojima, K. Okada, and M. Inaba. Humanmimic: Learning natural locomotion and transitions for humanoid robot via wasserstein adversarial imitation. *arXiv preprint arXiv:2309.14225*, 2023.

[270] T. Tao, M. Wilson, R. Gou, and M. van de Panne. Learning to get up. *arXiv preprint arXiv:2205.00307*, 2022.

[271] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

[272] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, and P. Fua. Structured prediction of 3d human pose with deep neural networks. *ArXiv*, abs/1605.05180, 2016.

[273] P. Tendulkar, D. Surís, and C. Vondrick. Flex: Full-body grasping without full-body grasps. In *CVPR*, 2023.

[274] C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[275] C. Tessler, Y. Kasten, Y. Guo, S. Mannor, G. Chechik, and X. B. 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.

[276] G. Tevet, S. Raab, S. Cohan, D. Reda, Z. Luo, X. B. Peng, A. H. Bermano, and M. van de Panne. Closd: Closing the loop between simulation and diffusion for multi-task character control. *arXiv preprint arXiv:2410.03441*, 2024.

[277] G. Tevet, S. Raab, B. Gordon, Y. Shafir, A. H. Bermano, and D. Cohen-Or. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[278] D. Tirumala, M. Wulfmeier, B. Moran, S. Huang, J. Humplik, G. Lever, T. Haarnoja, L. Hasenclever, A. Byravan, N. Batchelor, et al. Learning robot soccer from egocentric vision with deep reinforcement learning. *arXiv preprint arXiv:2405.02425*, 2024.

[279] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. Ieee, Oct. 2012.

[280] D. Tome, T. Alldieck, P. Peluse, G. Pons-Moll, L. Agapito, H. Badino, and F. De la Torre. Selfpose: 3d egocentric pose estimation from a headset mounted camera. *arXiv preprint arXiv:2011.01519*, 2020.

[281] D. Tome, P. Peluse, L. Agapito, and H. Badino. xr-egopose: Egocentric 3d human pose from an hmd camera. *arXiv preprint arXiv:1907.10045*, pages 7728–7738, Oct. 2019.

[282] Unitree. Unitree's first universal humanoid robot, 2023.

[283] A. Van Den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. *Adv. Neural Inf. Process. Syst.*, 2017-Decem:6307–6316, 2017.

[284] M. Vondrak, L. Sigal, J. Hodgins, and O. C. Jenkins. Video-based 3d motion capture through biped control. *ACM Transactions on Graphics (TOG)*, 31:1–12, 2012.

[285] N. Wagener, A. Kolobov, F. V. Frujeri, R. Loynd, C.-A. Cheng, and M. Hausknecht. Mocapact: A multi-task dataset for simulated humanoid control. *arXiv preprint arXiv:2208.07363*, 2022.

[286] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[287] W. Wan, H. Geng, Y. Liu, Z. Shan, Y. Yang, L. Yi, and H. Wang. Unidexgrasp++: Improving dexterous grasping policy learning via geometry-aware curriculum and iterative generalist-specialist learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3891–3902, 2023.

[288] J. Wang, L. Liu, W. Xu, K. Sarkar, D. Luvizon, and C. Theobalt. Scene-aware egocentric 3d human pose estimation. *arXiv preprint arXiv:2212.11684*, 2022.

[289] J. Wang, L. Liu, W. Xu, K. Sarkar, and C. Theobalt. Estimating egocentric 3d human pose in global space. *arXiv preprint arXiv:2104.13454*, 2021.

[290] J. Wang, Z. Luo, Y. Yuan, Y. Li, and B. Dai. Pacer+: On-demand pedestrian animation controller in driving scenarios. *arXiv preprint arXiv:2404.19722*, 2024.

[291] J. Wang, Y. Yuan, Z. Luo, K. Xie, D. Lin, U. Iqbal, S. Fidler, S. Khamis, H. Kong, and Mellon University, Carnegie. Learning human dynamics in autonomous driving scenarios. International Conference on Computer Vision, 2023, 2023.

[292] T. Wang, Y. Guo, M. Shugrina, and S. Fidler. Unicon: Universal neural controller for physics-based character motion. 2020.

[293] Y. Wang, J. Lin, A. Zeng, Z. Luo, J. Zhang, and L. Zhang. Physhoi: Physics-based imitation of dynamic human-object interaction. *arXiv preprint arXiv:2312.04393*, 2023.

[294] Y. Wang, Q. Zhao, R. Yu, A. Zeng, J. Lin, Z. Luo, H. W. Tsui, J. Yu, X. Li, Q. Chen, et al. Skillmimic: Learning reusable basketball skills from demonstrations. *arXiv preprint arXiv:2408.15270*, 2024.

[295] A. Winkler, J. Won, and Y. Ye. Questsim: Human motion tracking from sparse sensors with simulated avatars. *arXiv preprint arXiv:2209.09391*, 2022.

[296] J. Won, D. Gopinath, and J. Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4):33–1, 2020.

[297] J. Won, D. Gopinath, and J. Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Trans. Graph.*, 39, 2020.

[298] J. Won, D. Gopinath, and J. Hodgins. Control strategies for physically simulated characters performing two-player competitive sports. *ACM Trans. Graph.*, 40:1–11, 2021.

[299] J. Won, D. Gopinath, and J. Hodgins. Physics-based character controllers using conditional vaes. *ACM Trans. Graph.*, 41:1–12, 2022.

[300] Y. Wu, K. Karunratanakul, Z. Luo, and S. Tang. Uniphys: Unified planner and controller with diffusion for flexible physics-based character control. *arXiv preprint arXiv:2504.12540*, 2025.

[301] Y. Wu, J. Wang, Y. Zhang, S. Zhang, O. Hilliges, F. Yu, and S. Tang. Saga: Stochastic whole-body grasping with contact. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[302] Y. Wu and K. He. *Group Normalization*, pages 3–19. Lecture notes in computer science. Springer International Publishing, 2018.

[303] K. Xie, T. Wang, U. Iqbal, Y. Guo, S. Fidler, and F. Shkurti. Physics-based human motion estimation and synthesis from videos. *arXiv preprint arXiv:2109.09913*, 2021.

[304] X. Xie, B. L. Bhatnagar, and G. Pons-Moll. Visibility aware human-object interaction tracking from single rgb camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4757–4768, 2023.

[305] Z. Xie, J. Tseng, S. Starke, M. van de Panne, and C. K. Liu. Hierarchical planning and control for box loco-manipulation. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 6(3):1–18, 2023.

[306] K. Xu, Z. Hu, R. Doshi, A. Rovinsky, V. Kumar, A. Gupta, and S. Levine. Dexterous manipulation from images: Autonomous real-world rl via substep guidance. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5938–5945. IEEE, 2023.

[307] M. Xu, Z. Xu, Y. Xu, C. Chi, G. Wetzstein, M. Veloso, and S. Song. Flow as the cross-domain manipulation interface. *arXiv preprint arXiv:2407.15208*, 2024.

[308] S. Xu, H. Y. Ling, Y.-X. Wang, and L.-Y. Gui. Intermimic: Towards universal whole-body control for physics-based human-object interactions, 2025.

[309] W. Xu, A. Chatterjee, M. Zollhoefer, H. Rhodin, P. Fua, H.-P. Seidel, and C. Theobalt. Mo2cap2: Real-time mobile 3d motion capture with a cap-mounted fisheye camera. *arXiv preprint arXiv:1803.05959*, 2018.

[310] W. Xu, A. Chatterjee, M. Zollhoefer, H. Rhodin, P. Fua, H.-P. Seidel, and C. Theobalt. Mo 2 cap 2: Real-time mobile 3d motion capture with a cap-mounted fisheye camera. *IEEE transactions on visualization and computer graphics*, 25(5):2093–2101, 2019.

[311] X. Xu, Y. Zhang, Y.-L. Li, L. Han, and C. Lu. Humanvla: Towards vision-language directed object rearrangement by physical humanoid. *Advances in Neural Information Processing Systems*, 37:18633–18659, 2025.

[312] Y. Xu, W. Wan, J. Zhang, H. Liu, Z. Shan, H. Shen, R. Wang, H. Geng, Y. Weng, J. Chen, et al. Unidexgrasp: Universal robotic dexterous grasping via learning diverse proposal generation and goal-conditioned policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4737–4746, 2023.

[313] Y. Xu, S. Zhu, and T. Tung. Denserac: Joint 3d pose and shape estimation by dense render-and-compare. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7759–7769, 2019.

[314] K. Yamane, S. O. Anderson, and J. K. Hodgins. Controlling humanoid robots with human motion data: Experimental validation. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pages 504–510. IEEE, 2010.

[315] F. Yang, C. Ma, J. Zhang, J. Zhu, W. Yuan, and A. Owens. Touch and go: Learning from human-collected vision and touch. *arXiv preprint arXiv:2211.12498*, 2022.

[316] L. Yang, K. Li, X. Zhan, F. Wu, A. Xu, L. Liu, and C. Lu. OakInk: A large-scale knowledge repository for understanding hand-object interaction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[317] Y. Yang, G. Shi, X. Meng, W. Yu, T. Zhang, J. Tan, and B. Boots. Cajun: Continuous adaptive jumping using a learned centroidal controller. In *Conference on Robot Learning*, pages 2791–2806. PMLR, 2023.

[318] H. Yao, Z. Song, B. Chen, and L. Liu. Controlvae: Model-based learning of generative controllers for physics-based characters. *arXiv preprint arXiv:2210.06063*, 2022.

[319] Y. Ye, L. Liu, L. Hu, and S. Xia. Neural3points: Learning to generate physically realistic full-body motion for virtual reality users. In *Computer Graphics Forum*, volume 41, pages 183–194. Wiley Online Library, 2022.

[320] Y. Ye, A. Gupta, K. Kitani, and S. Tulsiani. G-hop: Generative hand-object prior for interaction reconstruction and grasp synthesis. In *CVPR*, 2024.

[321] Y. Ye, A. Gupta, and S. Tulsiani. What's in your hands? 3d reconstruction of generic objects in hands. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3895–3905, 2022.

[322] Y. Ye, X. Li, A. Gupta, S. De Mello, S. Birchfield, J. Song, S. Tulsiani, and S. Liu. Affordance diffusion: Synthesizing hand-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22479–22489, 2023.

[323] Y. Ye and C. K. Liu. Synthesis of detailed hand manipulations using contact sampling. *TOG*, 31(4):1–10, 2012.

[324] Y. Yuan and K. Kitani. 3d ego-pose estimation via imitation learning. In *Computer Vision – ECCV 2018*, volume 11220 LNCS, pages 763–778. Springer International Publishing, 2018.

[325] Y. Yuan and K. Kitani. Ego-pose estimation and forecasting as real-time pd control. *Proceedings of the IEEE International Conference on Computer Vision*, 2019-Octob:10081–10091, 2019.

[326] Y. Yuan and K. Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. *Lect. Notes Comput. Sci.*, 12354 LNCS:346–364, 2020.

[327] Y. Yuan and K. Kitani. Residual force control for agile human behavior imitation and extended motion synthesis. *arXiv preprint arXiv:2006.07364*, 2020.

[328] Y. Yuan, J. Song, U. Iqbal, A. Vahdat, and J. Kautz. Physdiff: Physics-guided human motion diffusion model. 2023.

[329] Y. Yuan, S.-E. Wei, T. Simon, K. Kitani, and J. Saragih. Simpoe: Simulated character control for 3d human pose estimation. *CVPR*, 2021.

[330] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. *The International Journal of Robotics Research*, 41(7):690–705, 2022.

[331] H. Zhang, Y. Yuan, V. Makoviychuk, Y. Guo, S. Fidler, X. B. Peng, and K. Fatahalian. Learning physically simulated tennis skills from broadcast videos. *ACM Trans. Graph.*, 42:1–14, 2023.

[332] H. Zhang, Y. Ye, T. Shiratori, and T. Komura. Manipnet: neural manipulation synthesis with a hand-object spatial representation. *TOG*, 40(4):1–14, 2021.

[333] H. Zhang, J. Cao, G. Lu, W. Ouyang, and Z. Sun. Learning 3d human shape and pose from dense body parts. *ArXiv*, abs/1912.13344, 2019.

[334] H. Zhang, S. Christen, Z. Fan, O. Hilliges, and J. Song. GraspXL: Generating grasping motions for diverse objects at scale. *arXiv preprint arXiv:2403.19649*, 2024.

[335] H. Zhang, S. Christen, Z. Fan, L. Zheng, J. Hwangbo, J. Song, and O. Hilliges. ArtiGrasp: Physically plausible synthesis of bi-manual dexterous grasping and articulation. In *I3DV*, 2024.

[336] J. Y. Zhang, P. Felsen, A. Kanazawa, and J. Malik. Predicting 3d human dynamics from video. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7114–7123, 2019.

[337] J. Z. Zhang, S. Yang, G. Yang, A. L. Bishop, S. Gurumurthy, D. Ramanan, and Z. Manchester. Slomo: A general system for legged robot motion imitation from casual videos. *IEEE Robotics and Automation Letters*, 2023.

[338] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022.

[339] Y. Zhang, D. Huang, B. Liu, S. Tang, Y. Lu, L. Chen, L. Bai, Q. Chu, N. Yu, and W. Ouyang. Motiongpt: Finetuned llms are general-purpose motion generators. *arXiv preprint arXiv:2306.10900*, 2023.

[340] Y. Zhang, A. Clegg, S. Ha, G. Turk, and Y. Ye. Learning to transfer in-hand manipulations using a greedy shape curriculum. In *Computer Graphics Forum*, volume 42, pages 25–36. Wiley Online Library, 2023.

[341] Y. Zhang, D. Gopinath, Y. Ye, J. Hodgins, G. Turk, and J. Won. Simulation and retargeting of complex multi-character interactions. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–11, 2023.

[342] D. Zhao, Z. Wei, J. Mahmud, and J.-M. Frahm. Egoglass: Egocentric-view human pose estimation from an eyeglass frame, 2021.

[343] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[344] K. Zhou, B. L. Bhatnagar, J. E. Lenssen, and G. Pons-Moll. Toch: Spatio-temporal object-to-hand correspondence for motion refinement. In *ECCV*. Springer, October 2022.

[345] Y. Zhou, T. Lei, H. Liu, N. Du, Y. Huang, V. Zhao, A. Dai, Z. Chen, Q. Le, and J. Laudon. Mixture-of-experts with expert choice routing. *arXiv preprint arXiv:2202.09368*, 2022.

[346] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:5738–5746, 2019.

[347] Q. Zhu, H. Zhang, M. Lan, and L. Han. Neural categorical priors for physics-based character control. *arXiv preprint arXiv:2308.07200*, 2023.

[348] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. *arXiv preprint arXiv:2309.05665*, 2023.

[349] Y. Zou, J. Yang, D. Ceylan, J. Zhang, F. Perazzi, and J.-B. Huang. Reducing footskate in human motion reconstruction with ground contact constraints. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2020.