Learning Whole-body Manipulation for Quadrupedal Robot

Seunghun Jeon[1†], Moonkyu Jung[1†], Suyoung Choi[1†], Beomjoon Kim[*2†] and Jemin Hwangbo[*1†]

***Abstract*— We propose a learning-based system for enabling quadrupedal robots to manipulate large, heavy objects using their whole body. Our system is based on a hierarchical control strategy that uses the deep latent variable embedding which captures manipulation-relevant information from interactions, proprioception, and action history, allowing the robot to implicitly understand object properties. We evaluate our framework in both simulation and real-world scenarios. In the simulation, it achieves a success rate of 93.6 $\%$ in accurately re-positioning and re-orienting various objects within a tolerance of 0.03 m and 5 $^\circ$. Real-world experiments demonstrate the successful manipulation of objects such as a 19.2 kg water-filled drum and a 15.3 kg plastic box filled with heavy objects while the robot weighs 27 kg. Unlike previous works that focus on manipulating small and light objects using prehensile manipulation, our framework illustrates the possibility of using quadrupeds for manipulating large and heavy objects that are ungraspable with the robot's entire body. Our method does not require explicit object modeling and offers significant computational efficiency compared to optimization-based methods. The video can be found at https://youtu.be/fO_PVr27QxU.**

***Index Terms*— Reinforcement Learning, Legged Robots, Deep Learning Methods**

I. Introduction

While recent quadrupedal robots have a remarkable capability to maneuver through a diverse set of challenging and complex terrains [1]–[3], they still lack the ability to handle tasks requiring interaction with objects and the environment. There have been attempts to attach manipulators to quadrupedal robots [4]–[7]; however, in industrial environments, quadrupeds often face situations where they need to move objects that are large, heavy, and have agnostic physical properties (e.g. shape, mass, inertia, COM). Consequently, prehensile manipulation in such an environment becomes challenging. Our goal is to endow quadrupeds with the capability to manipulate such objects from high-dimensional sensory data using their entire body. This is a challenging task that involves not only determining torque commands to balance and locomote amidst contact with the object but also reasoning about the physical properties of objects to determine a long sequence of actions that would move the object to the target location.

The conventional approach to whole-body manipulation involves designing an optimization-based planner and controller by carefully modeling the dynamics of the object, environment, and robot [5], [8]–[11]. However, this approach has two drawbacks. Firstly, it incurs a significant computational cost due to hybrid dynamics and decision variables involving binary contact sequences and continuous motions under several dynamics constraints that change with respect to contact decisions. With simplifying assumptions, we can significantly reduce the computational cost, but they come at the expense of limited robot motion types [8], [9]. Secondly, these models rely on the accurate estimation of explicit object information, such as its shape and physical parameters, from sensory data, and the action history itself is a challenging task.

Recently, learning-based methods for whole-body manipulation have been proposed [12]–[16]. Model-free approaches offer advantages over conventional optimization-based methods by enabling robots to adapt to diverse environments using the generalization capability of neural network (NN) [13], [15]. Additionally, the computation of the actions comes down to making predictions from a NN, instead of a complex optimization procedure, significantly reducing the computational cost. However, previous works have primarily focused on training systems to manipulate a single object, such as a ball [13] or a cylinder [15], which introduces inherent challenges when dealing with objects of various types, geometries, and physical properties. The difficulty arises from the need to adjust forces depending on the specific object type.

In this paper, we tackle the problem with a hierarchical control strategy. The high-level controller outputs velocity commands based on the state of the robot and the object, while the low-level controller outputs desired joint position that leads joint torque to track given commands. Additionally, we show that the difficulties mentioned above — manipulating a wide range of objects with diverse physical properties— can be mitigated by using a deep latent variable encoding model that embeds object information from multiple interactions, proprioception, and action history into a low-dimensional latent vector. This enables an implicit understanding of object physical properties, akin to how humans manipulate an object when they lack knowledge about the object physical properties. Initially, humans rely on prior knowledge to determine the force's direction and magnitude. Subsequently, through physical interactions, they refine their understanding, considering factors like the applied force vector and perceived object motion.

We conduct extensive evaluations, achieving a success rate of 93.6 $\%$ in re-positioning various objects to the desired pose within 0.03 m and 5° in simulation. Moreover, we successfully deployed the learning-based system, trained in

This work was supported in part by Korea Evaluation Institute of Industrial Technology (KEIT) funded by the Korea Government (MOTIE) under Grant No.20018216.

† Korea Advanced Institute of Science and Technology (KAIST)

∗ corresponding author

[1] Robot Artificial Intelligence Laboratory [2]Intelligent Mobile-Manipulation Laboratory

jhwangbo@kaist.ac.kr, beomjoon.kim@kaist.ac.kr

Fig. 1: **Whole-body manipulation for planar pushing**: The proposed system enables a quadrupedal robot, weighing 27 kg, to manipulate its entire body to re-pose an object in a 2D plane. Our system does not require the object's physical properties or dimensions but achieves an implicit understanding of the object's properties through real-time interactions. Experiments were conducted on a total of five different objects with diverse properties. The robot can adapt even when the object's physical properties change dynamically, such as when dealing with a 19.2 kg water-filled drum that is approximately 70 $\%$ of the robot's weight.

simulation, directly to the real-world scenario without any fine-tuning or additional training. The results in the real world, show that our proposed system can manipulate heavy and complex objects without a single failure, such as water-filled drums weighing 19.2 kg and plastic boxes filled with wooden furniture weighing 15.3 kg, as shown in Figure. 1.

II. Related Work

A. Whole-body manipulation for legged robots

Rigo et al. [8] propose a real-time controller for non-prehensile loco-manipulation that tracks a given object trajectory under the given dynamics and assumes canonical push dynamics. The proposed method could re-position a 5 kg box-shaped object to the desired pose by optimizing contact point location and force magnitude and consequently derive desired joint torques via a hierarchical Model Predictive Control (MPC) framework. Sombolestan et al. [9] present an approach for whole-body manipulation tasks on unknown objects and terrains without prior knowledge. They propose a real-time adaption method that can push an object where physical properties vary with time weighing up to 7 kg, even on slopes in a single direction. These works demonstrate the potential for optimization-based approaches to enable robots to perform loco-manipulation for pushing. However, these approaches assume a single, flat-surface contact at the pre-defined robot's head, not the whole body of the robot.

Learning-based approaches also have been considered in whole-body manipulation. Ji et al. [13] successfully overcome challenges related to perceiving and controlling a ball on different terrains. The robot action policy is trained to achieve desired object-centric velocity command under assorted surroundings via Reinforcement Learning (RL). Kumar et al. [15] presents a cascaded compositional residual learning approach for complex interactive behaviors. The approach decomposes a complex behavior into a sequence of simpler sub-behaviors and learns a residual policy for each sub-behavior. The residual policies are then combined to form a policy for the complex behavior. They evaluate their approach to a variety of tasks, including cylindrical object pushing. In this task, the policy was able to successfully push the object within 0.1 m of the target location 89 $\%$ of the time. These works demonstrate a learning-based approach can be successfully applied to dynamic whole-body manipulation. However, previous studies have not considered generalization over a variety of objects, and they have not considered situations where the physical properties of objects change significantly in real-time, or where the goal includes orientation. We address these aspects in this work.

B. Object system identification through physical interactions

Various methods have been developed for object system identification through physical interactions, aiming to estimate the object parameters and, consequently, the motion.

Wang et al. [17] propose SwingBot, which utilizes tactile feedback data collected during the execution of predefined motions to implicitly encode physical properties. Kloss et al. [18] employs the Kalman filter to estimate object physical properties, including the center of mass, friction, and mass. Song et al. [19] propose a probabilistic model for identifying the mass and friction properties of objects with unknown material properties, which is trained on a simulated system and is effective in real-world experiments. Mavrakis et al. [20] propose a data-driven method for estimating the object's inertial parameters, which is trained on a large dataset of simulated and real-world pushes. Xu et al. [21] introduce DensePhysNet, a neural network-based approach to implicitly represent object physical properties. Fragkiadaki et al. [22] predict ball motion under a pushing force using object-centric prediction approaches. These methods either explicitly estimate physical parameters or implicitly encode them using neural networks, contributing to the understanding and prediction of object motion based on real-world experiences. Our system involves predicting object properties implicitly through a neural network using physical interactions.

C. Privileged learning and adaptation

Many studies employ a simulation-to-real adaptation approach, where the policy is trained using privileged information (i.e. teacher policy) that is only observable in simulation and remains agnostic to the real-world scenario. This enables the policy to quickly achieve high performance when deployed in the real world. In some studies, the vision sensory data, robot, and environment's physical properties are implicitly estimated [1], [14], [23]. For instance, Lee et al. [1] distilled the teacher policy's action output after training through imitation learning. Kumar et al. [23] trained the privileged information encoder and adapted the encoded latent feature space via 2-step training. Fu et al. [14] remove the two-phase teacher-student scheme, and regularize environment extrinsics latent to avoid large deviations. While others explicitly estimated the privileged information, Ji et al. [24] concurrently trained the state estimator for the base linear velocity, foot height, and contact probability. Additionally, some studies [25], [26] use an asymmetric structure, where the critic evaluates the value only through privileged information, while the actor makes actual decision-making based on observable values. We leverage privileged learning in an implicit latent space that encodes an object physical properties.

III. Learning whole-body manipulation for planar pushing

Our task is to determine a sequence of robot joint torques that manipulates an arbitrary object to the target pose using the robot's whole body. We assume that we do not have prior knowledge about the physical properties of the object, but have complete knowledge of the object's and robot's poses. We assure, at the start, there is a considerable distance between the object and the robot, so the robot first has to approach the object and push it to the target pose. Our system has a hierarchical structure, as illustrated in Figure 2. In the following sections, we provide a detailed description of each component of our system.

A. Training whole-body manipulation in simulation

TABLE I: **Notations**: Variable representation

Notations		Components	Dim	Components	Dim	Total
s_t	o_t	$\phi_{robot,x}$	3	v_{robot}	3	33
		w_{robot}	3	$\phi_{robot}^T(p_{left\_foot} - p_{robot})$	3	
		$\phi_{robot}^T(p_{right\_foot} - p_{robot})$	3	$\frac{p_{object} - p_{robot}}{\|\|p_{object} - p_{robot}\|\|_2}$	2	
		$\|\|p_{object} - p_{robot}\|\|_2$	1	$\frac{p_{target} - p_{object}}{\|\|p_{target} - p_{object}\|\|_2}$	2	
		$\|\|p_{target} - p_{object}\|\|_2$	1	$\frac{p_{target} - p_{robot}}{\|\|p_{target} - p_{robot}\|\|_2}$	2	
		$\|\|p_{target} - p_{robot}\|\|_2$	1	$\phi_{robot}{}^T v_{object}$	3	
		$\phi_{robot,x}{}^T \phi_{object,x}$	1	$\|\|\phi_{robot,x} \times \phi_{object,x}\|\|_2$	1	
		$\phi_{robot,x}{}^T \phi_{target,x}$	1	$\|\|\phi_{robot,x} \times \phi_{target,x}\|\|_2$	1	
		$\phi_{object,x}{}^T \phi_{target,x}$	1	$\|\|\phi_{object,x} \times \phi_{target,x}\|\|_2$	1	
	x_t	$Object\ type$	3	$Dimension$	3	22
		m_{object}	1	COM_{object}	3	
		I_{object}	9	$Friction\ coeff$	1	
		$Drag\ coeff$	1	$Contact$	1	
	n_t	$\phi_{robot,z}$	3	w_{robot}	3	33
		v_{robot}	3	q	12	
		$\dot{q}$	12			

We first define the variables as shown in Table I for the following descriptions. The state s_t is the concatenation of observation o_t, privileged information x_t, and robot state n_t. Here, $\phi_{\langle\cdot\rangle}$ represents the rotation matrix, and $\phi_{\langle\cdot\rangle,x}$ represents the x-axis component of the rotation matrix (i.e. first column vector). $v_{\langle\cdot\rangle}$ represents the COM velocity, while $w_{\langle\cdot\rangle}$ represent the COM angular velocity. The symbol $\times$ corresponds to the cross product. The variable $p_{\langle\cdot\rangle}$ represents the geometric center position. The object type is represented as a one-hot vector, and the dimension information is represented as {width, depth, height} of the object. $m_{\langle\cdot\rangle}$ represents the mass, and $I_{\langle\cdot\rangle}$ represents the inertia. $Contact$ represents robot actually contacts the object or not. Additionally, q represents the robot joint position, and $\dot{q}$ represents the robot joint velocity.

a) Low-level controller training: The low-level controller, denoted as $\pi^l : \mathcal{N} \times \mathcal{A} \to \mathcal{T}$, predicts the joint position target $\boldsymbol{q}_t^{des} \sim \pi^l(\boldsymbol{q}_t^{des}|n_t, a_t) \in \mathbb{R}^{12}$, and converts it to robotic torque command $\boldsymbol{\tau}$ through the joint impedance controller. The low-level controller internally has an explicit estimator that predicts the body velocity $v \in \mathbb{R}^3$, feet height $z_{feet} \in \mathbb{R}^4$, and body height $z_{body} \in \mathbb{R}^1$. The learning architecture and rewards are the same as those in [3]. The low-level controller's role is to track the given commands $a_t = [v_x^{cmd}, v_y^{cmd}, w_{yaw}^{cmd}] \in se(2)$, which consist of desired base linear velocities in the forward and lateral directions, as well as the desired yaw rate. During the training of the low-level controller, velocity commands a_t are randomly sampled in every iteration.

b) High-level controller training: The high-level controller consists of two distinct elements: the high-level policy, $\pi_\theta^h : \mathcal{L} \times \mathcal{O} \times \mathcal{A} \to \mathcal{A}$, and the encoder $h_\psi : \mathcal{X} \to \mathcal{L}$ (in simulation) or $h_\phi : \mathcal{O} \times \mathcal{A} \to \mathcal{L}$ (in real-world). h_ψ takes the privileged history $X_t \in \mathbb{R}^{22 \times N}$, consisting of $N = 20$ privileged information history $x_t \in \mathbb{R}^{22}$, and predicts the latent vector sequence $\mathbf{l}_t^{1:N} \in \mathbb{R}^{96 \times N}$. The latest encoded latent vector $\mathbf{l}_t^N$, current observation o_t, previous action a_{t-1}

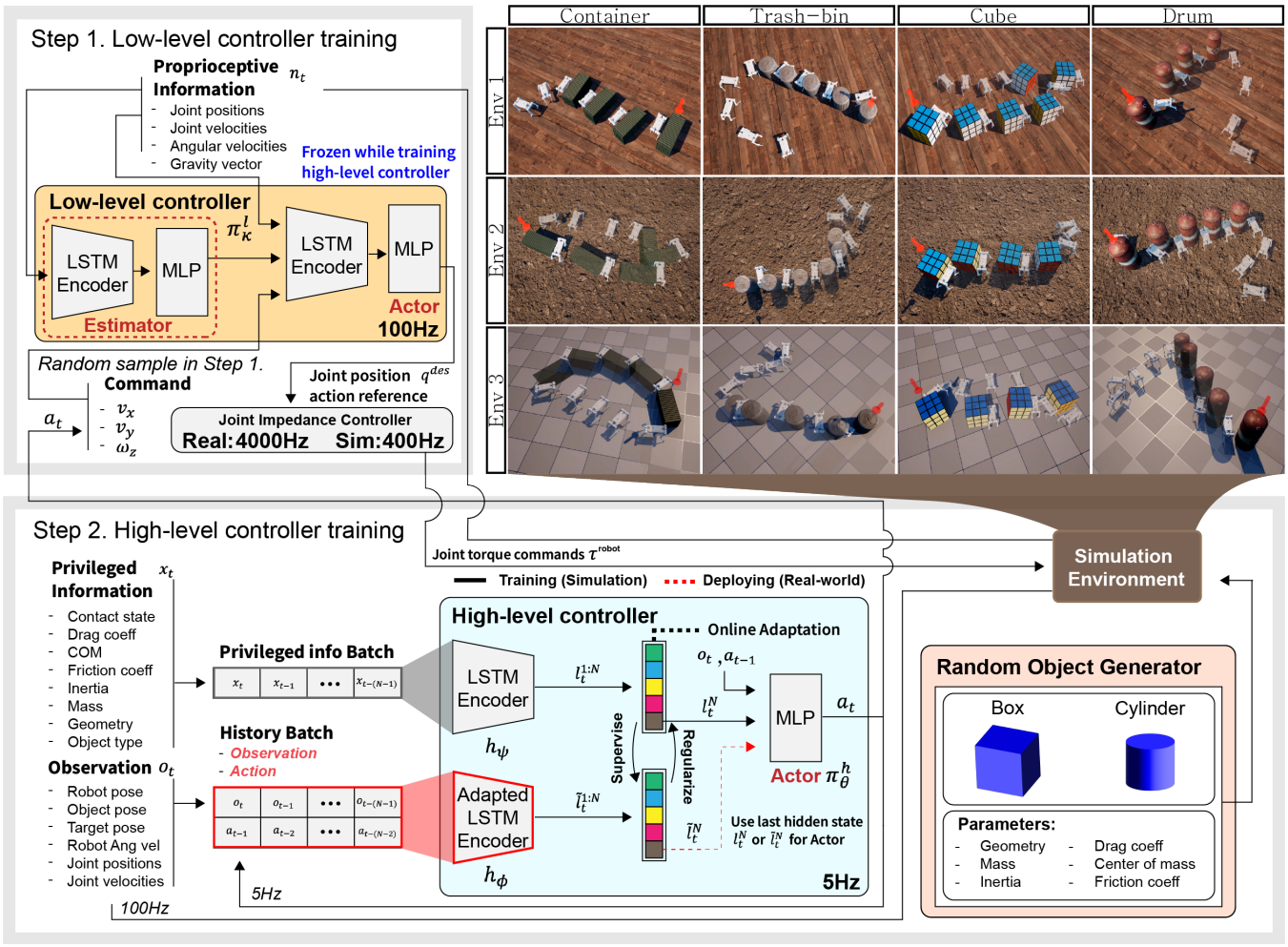


Fig. 2: **Overall framework**: In step 1, we train the low-level controller alone via reinforcement learning. The low-level controller tracks a given velocity command, following the method in [3]. In step 2, we freeze the low-level controller and only train the high-level controller. The high-level policy outputs an action command from a current observation o_t, previous action a_{t-1}, and a latent vector that encodes a history batch of privileged information x_t. It is also trained via reinforcement learning. Adaptation is performed online on the encoder of the high-level controller through supervised learning and regularization of latent vectors, following the method in [14]. The high-level controller generates the robot body $se(2)$ velocity every 5Hz, and this action command is fed to the low-level controller. The low-level controller generates the desired joint positions at 100Hz, which are converted to torque via an impedance controller.

are inputs for the high-level policy π^h to predict the action $a_t \in se(2)$. The predicted action a_t is provided at intervals of 0.2 seconds, and the low-level controller π^l tracks the action command a_t at a frequency of 0.01 seconds. To adhere to the action limits, the actions are constrained using the softmax function, limiting v_x^{cmd}, v_y^{cmd}, and w_{yaw}^{cmd} to the range of (-1.5, 1.5) m/s, (-1.5, 1.5) m/s, and (-1.5, 1.5) rad/s, respectively. During the training of the high-level controller, the pre-trained low-level controller remains frozen.

$$\mathbf{l}_t^{1:N} = h_\psi(X_t) \tag{1}$$
$$a_t \sim \pi_\theta^h(a_t|\mathbf{l}_t^N, o_t, a_{t-1}) \tag{2}$$
$$\boldsymbol{\tau}^{robot} \sim \pi^l(\boldsymbol{\tau}^{robot}|n_t, a_t) \tag{3}$$
$$X_t = \{x_t^1, ..., x_t^{N-1}, x_t^N\} \tag{4}$$

We implement both h_ψ and h_ϕ as a Long Short Term Memory (LSTM), and π_θ^h as Multi-Layer Perceptrons (MLPs). We concurrently train the encoder h_ψ and the high-level policy π_θ^h using end-to-end model-free reinforcement learning. The reinforcement learning objective is to maximize the expected return, defined as the sum of discounted rewards over time steps t:

$$J(\pi^h, \pi^l, h_\psi) = \mathbb{E}_{\tau \sim p(\tau|\pi^h, \pi^l, h_\psi)}\left[\sum_{t=0}^{T-1} \gamma^t r_t\right] \tag{5}$$

c) RL Reward function: In the planar pushing task, the main objective is to minimize the distance between the object and the goal within the $SE(2)$ manifold. However, finding the optimal weight between rotation and position components can be challenging and may result in sub-optimal perfor-

TABLE II: Reward Functions

Reward		Expression			
r_1^i		$k_1 \exp(-\|\|p_{\text{robot}} - p_{\text{object}}\|\|_2)$			
r_2^e		$k_2 \log(\|\|G_{\text{goal}} - G_{\text{object}}\|\|_2 + 0.05)$			
r_3^i		$k_3 \exp(-\phi^T_{\text{robot},x} v_{\text{object}})$			
r_4^i		$k_4 \exp(-(p_{\text{goal}} - p_{\text{object}})^T v_{\text{object}})$			
r_5^i		$k_5 \exp(-\|\|a_t - a_{t-1}\|\|_2)$			
r_6^i		$k_6 \exp(-\|\|a_t - 2a_{t-1} + a_{t-2}\|\|_2)$			
Intrinsic Reward					
$r_t^i = \begin{cases} r_{t-1}^i & \text{if } \|\|p_{\text{goal}} - p_{\text{object}}\|\|_2 < 0.2\ m \\ k_7(r_1^i + r_3^i + r_4^i + r_5^i + r_6^i) & \text{otherwise} \end{cases}$					
Extrinsic Reward		$r_t^e = k_8 r_2^e$			
Reward Coefficients					
k_1	1	k_2	4	k_3	1
k_4	1	k_5	2	k_6	3
k_7	1	k_8	1		

mance. To overcome this challenge, we directly minimize the distance between the key points of the object and the goal in the $\mathbb{R}^3$ space (i.e., extrinsic reward r^e) similar to [27], [28] as shown in the Figure. 3. $G_{\langle\cdot\rangle} \in \mathbb{R}^{24}$ represents the set of 8 key points position, all in the robot's base frame.

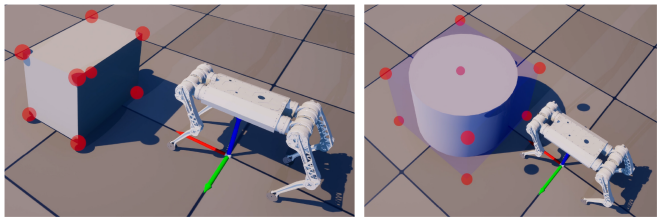


Fig. 3: **8 key points**: The 8 key points are defined as rectified of the bounding box that encapsulates the object.

However, if we only consider extrinsic reward, the agent gets stuck in local minima when it is not in contact with the object (e.g. during the approaching phase) during which extrinsic rewards do not change. To address this, we introduce intrinsic rewards r^i to encourage exploration to approach the target object. The intrinsic rewards consist of the following components: reducing the distance between the robot and the object (r_1^i), encouraging the object to be pushed in the robot's forward direction (r_3^i), inducing the object's movement direction towards the goal direction (r_4^i), and action smoothness rewards for smooth movements (r_5^i, r_6^i). The intrinsic reward is no longer updated (i.e. intrinsic reward maintains its previous value) once the robot is within a certain distance (0.2 m) of the object in order to prevent interrupting the extrinsic rewards to converge to the maximum. The total reward, which combines intrinsic and extrinsic rewards, is defined as $r_t^{total} = r_t^i + r_t^e$.

d) Transferring to the real world: The system cannot be directly deployed on the real robot because the privileged information x_t is not observable. To address this limitation, we use ROA (Regularized Online Adaptation) [14] for 1-step privileged learning. This method trains a teacher $h_\psi : \mathcal{X} \to \mathcal{L}$ and a student $h_\phi : \mathcal{H} \to \mathcal{L}$ simultaneously in online. The teacher ψ takes the privileged information history $X = \{x^1, ..., x^{N-1}, x^N\}$ as input and predicts a latent vector $\mathbf{l}$ to implicitly encode object properties and adapt the policy to various environments. Meanwhile, the student h_ϕ imitates the teacher h_ψ to approximate $\tilde{\mathbf{l}}$ based on the observation and action history $H = \{o^1, a^1, ..., o^{N-1}, a^{N-1}, o^N, a^N\}$. This enables the student to estimate the encoded object properties from observable data. Here, the stacking process is asynchronous: the observation and privileged information history is updated at the low-level policy frequency (100Hz), and the action history is updated at the high-level policy frequency (5Hz). This is because state transitions occur after every low-level policy execution. The teacher ψ and the student ϕ are trained simultaneously using reinforcement learning (5) with the following auxiliary loss:

$$L_{adaptation} = \lambda||\tilde{\mathbf{l}} - sg[\mathbf{l}]||_2 + ||sg[\tilde{\mathbf{l}}] - \mathbf{l}||_2 \tag{6}$$

where $sg[\cdot]$ is the stop gradient operator, and λ is the Lagrangian multiplier. The first term is for supervision, and the second term is for regularization. The entire high-level controller training procedure is presented in 1.

Algorithm 1: Learning the high-level controller in simulation (with fixed low-level controller)

for *episode=1, M* **do**
  Initialize h_ψ, h_ϕ
  Initialize action history A_{queue} to capacity N
  Initialize state history S_{queue} to capacity N
  Initialize data batch D
  /* High-level controller, 5Hz */
  for $t = 1, T$ **do**
    $H = \langle O_{history}, A_{queue} \rangle$ // $O_{history} \in S_{queue}$
    $\mathbf{l}_t^{1:N} = h_\psi(X_{history})$ // $X_{history} \in S_{queue}$
    $\tilde{\mathbf{l}}_t^{1:N} = h_\phi(H)$ // $\tilde{\tau} \in \tau$
    $a_t \sim \pi_\theta^h(a_t | l_t^N, o_t, a_{t-1})$
    $A_{queue} \leftarrow a_t$
    /* Low-level controller, 100Hz */
    for $t' = 1, K$ **do**
      Observe state $s_{t'}$, and stack $S_{queue} \leftarrow s_{t'}$
      $\boldsymbol{\tau}_{t'}^{robot} = \pi_\kappa^l(n_{t'}, a_t)$ // $n_{t'} \in s_{t'}$
      Execute $\boldsymbol{\tau}_{t'}^{robot}$
    Store transition $s_{t'}, a_t, r_{t'}, \mathbf{l}_t, \tilde{\mathbf{l}}_t$ in D
  /* $(\mathbf{l}, \tilde{\mathbf{l}}, r, s) \sim D$ */
  $L_{adaptation} = \lambda||\tilde{\mathbf{l}} - sg[\mathbf{l}]||_2 + ||sg[\tilde{\mathbf{l}}] - \mathbf{l}||_2$
  $J = \mathbb{E}_{\tau \sim D}\left[\sum_{t=1}^{T} \gamma^t r_{t'}\right]$
  $L_{total} = -J + L_{adaptation}$
  Compute $\nabla_{\theta,\psi,\phi} L_{total}$ and update // PPO [29]

e) Training Environment Generation: During the training high-level controller, we utilized 300 parallel environments in simulation to efficiently collect data. Each environment was equipped with a **Random Object Generator** that spawns objects with various shapes and physical parameters. Table III shows the training and testing ranges for the object's

TABLE III: Training and testing parameters in simulation

Parameters		Training Range	Testing Range
Mass (kg)		U(6, 18)	U(10, 20)
Inertia axis angle θ (degree)		U(-20, 20)	U(-20, 20)
COM volume V (%)		U(0,50)	U(0,50)
Friction		U(0.2,0.4)	U(0.15, 0.4)
Drag coefficient		U(0.3,0.7)	U(0.0, 1.0)
Object type		[Box, Cylinder]	[Box, Cylinder]
Object dimension	Diameter (m)	U(0.5, 1.5)	U(0.5, 1.5)
	Width (m)	U(0.5, 1.5)	U(0.5,1.5)
	Depth (m)	U(0.5, 1.5)	U(0.5, 1.5)
	Height (m)	U(0.35, 1.05)	U(0.35, 1.05)

physical parameters. All variables are sampled from uniform distributions denoted as U. The parameters are randomly sampled in every episode. The center of mass (COM) is randomly chosen within $V = 50\%$ of the object's volume, starting from the geometric center of mass (x_0, y_0, z_0). To introduce diversity in the object's inertia, we diagonalize the original inertia tensor I to obtain the principal moments of inertia (I_1, I_2, I_3) and principal axes (u_1, u_2, u_3). We then transform the principal axis by an angle θ. The objects generated in the simulation include cylinders and boxes, each represented as a one-hot vector to indicate their shape. The object dimension is represented as a real-valued vector with three components: for cylinders {diameter, diameter, height}, and {width, depth, height} for boxes. All the physical parameters, shape, and dimension information of the objects belong to the privileged information x_t.

IV. Experiment and Result

We evaluate the capabilities of our framework both in simulation and the real world. We measure the success rate in the following task: re-posing objects using a quadruped robot under various conditions, including different object shapes and inertial parameters. We also examine the effectiveness of our system by comparing its performance against several baseline methods in simulation.

The training process was conducted in Raisim [30] using an AMD Ryzen9 5950X CPU and an NVIDIA GeForce RTX 3070 GPU. The training involved 300 environments and 10,000 iterations, with each iteration taking approximately 20 seconds in simulation. The overall training process took around 15 hours. For the deployment on the real robot, the trained networks were implemented on an Intel NUC (4-core) PC embedded in the robot. The pose information of the object and the robot was obtained in real-time using the VICON vero v2.2 motion capture system shown in Figure. 4b.

A. Simulation Result

In the simulation experiment, we randomly generate 1000 different situations based on the testing parameters in Table III. We evaluate the performance of the system using five criteria, each with different tolerance distances (d) and angles (θ) for the desired pose, as detailed in Table IV. Both the objects and the robot are spawned randomly with each position and orientation within 4 m and between 0 to 2π with respect to the world frame. The desired pose for the object is sampled within a range of translation up to 4 m and orientation from 0 to 2π from the object's initial pose. A trial is considered successful if the robot manipulated the object to the desired pose for each criterion within 30 seconds. We assess the success rate and the average time taken by the successful trials. We compare the performance of our method with the following baselines:

- w/o Pre-trained controller: In order to validate the performance enhancement of our hierarchical control structure, we exclude a pre-trained low-level controller, and instead high-level controller outputs joint position target q_t^{des} directly.
- w/o Adaptation (Domain Randomization): We train the high-level controller without an encoder, and the encoded latent vector input $\mathbf{l}^N$ was removed. Instead, we randomize the physical properties of the objects during training.
- w/o 8 key points: To validate the effectiveness of the 8 key points, we replace the extrinsic reward with the sum of the L2 norms of position and orientation errors.
- MLP Encoder: To verify whether the recurrent structure better captures the physical properties information of objects and the environment, we replace the LSTM encoder with an MLP encoder.
- w/o Inertial parameters: To verify whether our latent embedding actually estimates object inertial parameters, we exclude the object inertial parameters, including mass, inertia, and center of mass (COM) from the privileged information x_t.
- w/o Intrinsic switch: To verify whether considering both extrinsic and intrinsic rewards, even when close to the goal, can lead to sub-optimal behavior, we trained without intrinsic switching.
- Expert: We use the true value of the privileged information x_t, and the encoder h_ψ in simulation.

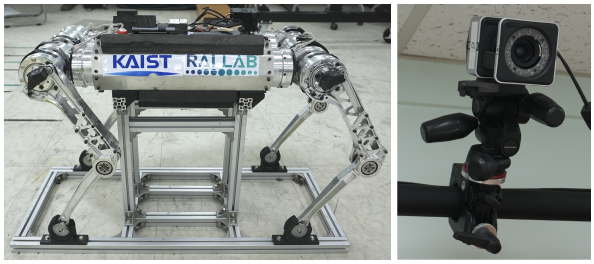
Table IV shows our simulation result. The proposed model shows slightly lower accuracy compared to the expert. The MLP encoder $h_{\langle\cdot\rangle}$ resulted in a loss of performance, indicating the difficulty in determining push behavior based solely on momentary data history. In the case without adaptation, the high-level policy demonstrates difficulties in decision-making, highlighting the importance of considering object physical properties inherently. When the pre-trained controller is not used, the robot failed to stand up or locomote properly, resulting in failure in all test cases. Additionally, without using the 8 key points, the performance significantly degraded, especially in terms of poor orientation alignment, as noted in criteria 1 to 3. Furthermore, the absence of an intrinsic switch led to decreased performance, particularly in criterion 5. This indicates that considering intrinsic rewards after a certain point can actually interfere with reaching the optimal behavior. Without considering inertial parameters, criterion 5 shows a severe performance loss, suggesting that the delicate manipulation of objects requires the consideration of their inertial parameters. This demonstrates that the proposed system effectively captures object

TABLE IV: Simulation Testing Results

Method	Success Rate (%)										Time (sec)	Observable
	Criteria 1		Criteria 2		Criteria 3		Criteria 4		Criteria 5			
	$d = 0.05m$	$\theta = 5°$	$d = 0.05m$	$\theta = 10°$	$d = 0.05m$	$\theta = 15°$	$d = 0.1m$	$\theta = 10°$	$d = 0.03m$	$\theta = 5°$		
w/o Pre-trained controller	0		0		0		0		0		-	✓
w/o Adaptation	58.4		56.1		60.9		67.4		49.8		18.3	✓
MLP Encoder	79.5		76.3		80.4		91.3		56.6		17.2	✓
w/o 8 key points	60.5		47.0		82.5		61.0		45.0		14.5	✓
w/o Intrinsic switch	91.6		91.0		91.6		92.9		82.5		13.2	✓
w/o Inertial parameters	84.5		82.6		85.1		89.3		68.2		17.3	✓
Ours	**96.1**		**96.1**		**96.3**		**96.8**		**93.6**		**11.23**	✓
Expert	97.8		97.8		97.9		98.0		96.7		10.5	-

inertial parameters via encoded latent vectors. Overall, our method outperformed the baseline methods, demonstrating its effectiveness in learning pushing behavior for a robot in a simulated environment.

B. *Real-world Experiment Result*



(a) Robot platform (b) Motion capture system

Fig. 4: **Real-world system**: (a) Raibo weights 27 kg and has a torque limit of 55 Nm, designed and built in-house. (b) VICON provides pose information for objects and robot at a frequency of 200Hz.

TABLE V: Real-world experiment object properties

Shape	Category	Mass (kg)	Dimension (m)
Box	Yellow	15.3	(Width,Depth,Height) = (0.45, 0.7, 0.5)
	Green	14.2	(Width,Depth,Height) = (0.4, 0.6, 0.45)
	Black	12.6	(Width,Depth,Height) = (0.3, 0.55, 0.65)
Cylinder	Trash bin	17.4	(Diameter,Height) = (0.56, 0.7)
	Drum	19.2	(Diameter,Height) = (0.4, 0.55)

For real-world experiments, we use the Raibo (Figure. 4a). We demonstrate the performance of our system on box and cylindrical shaped objects with various physical properties, including time-varying ones. Since the motion capture system area is limited, we fix the desired pose and the object's position and vary the object's orientation and the robot's pose. Three experiments were conducted for each object, with each initial orientation being 45°, 90°, or 180°. We define success as an object's final pose being within a certain tolerance, 0.05 m and 10°. The object properties used in the experiment are summarized in Table V. The objects were packed with furniture or water to change the center of mass and inertia in real-time during manipulation. In a total of 45 trials, (with 5 objects, 3 initial orientations, and 3 trials for each case) our system achieved without a single failure for all testing cases even though the robot had no prior

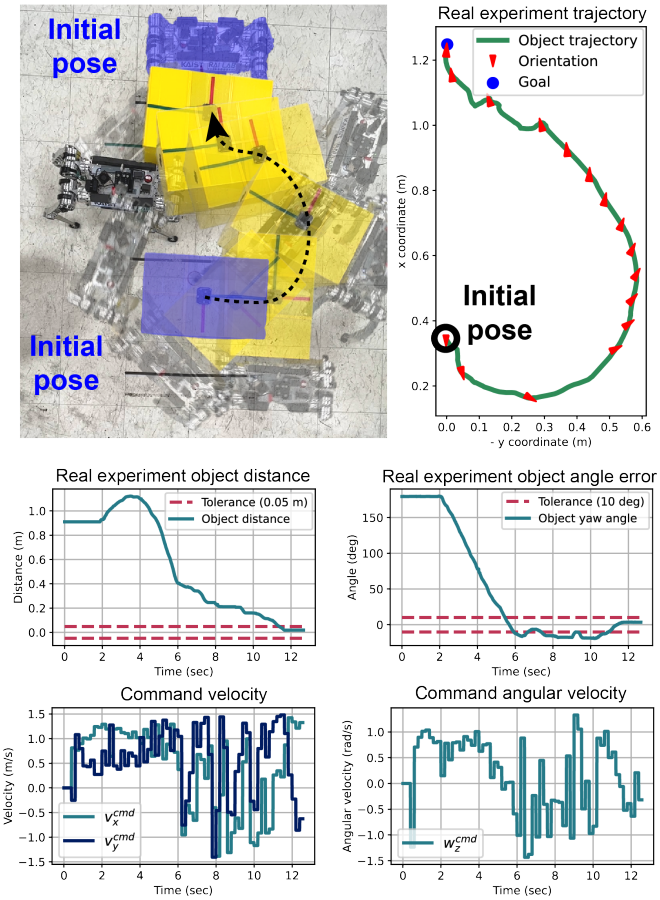


Fig. 5: **Real-world experiment result:** The figure shows the trajectory of the object and the $SE(2)$ error with respect to the goal pose. The command velocity predicted by the high-level controller during the experiment is shown below.

knowledge of the object's physical properties. We directly deployed our system in the real world without any fine-tuning (zero-shot learning). We present one of the real-world experiment results in Figure. 5.

V. Conclusion and Future Works

We have proposed a learning-based whole-body manipulation approach with a hierarchical structure for planar-pushing tasks. The proposed method allows the robot to manipulate objects without prior knowledge of their physical properties. Our system is computationally efficient, as it can perform

both approaching and pushing with NN predictions, without the need for trajectory optimization or contact reasoning. We have demonstrated the effectiveness of our approach through numerical and experimental validations. In simulations, we have shown that the proposed method achieves a remarkable success rate under diverse object physical properties. The system also demonstrates high accuracy in real-world robot experiments, manipulating objects up to 70 % of the robot's weight. We believe that this study could point to numerous directions for future research in developing more general methods for whole-body manipulation and loco-manipulation. Additionally, we are currently investigating methods for replacing the motion capture system with exteroceptive sensor data to avoid being restricted to a specific workspace. We are also interested in obstacle avoidance in cases where there are obstacles in the robot's path.

REFERENCES

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[2] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.

[3] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo, "Learning quadrupedal locomotion on deformable terrain," *Science Robotics*, vol. 8, no. 74, p. eade2256, 2023.

[4] G. Xin, F. Zeng, and K. Qin, "Loco-manipulation control for arm-mounted quadruped robots: Dynamic and kinematic strategies," *Machines*, vol. 10, no. 8, p. 719, 2022.

[5] 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)*. IEEE, 2019, pp. 8477–8483.

[6] Y. Ma, F. Farshidian, T. Miki, J. Lee, and M. Hutter, "Combining learning-based locomotion policy with model-based manipulation for legged mobile manipulators," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2377–2384, 2022.

[7] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter, "A unified mpc framework for whole-body dynamic locomotion and manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4688–4695, 2021.

[8] A. Rigo, Y. Chen, S. K. Gupta, and Q. Nguyen, "Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control," *arXiv preprint arXiv:2210.03442*, 2022.

[9] M. Sombolestan and Q. Nguyen, "Hierarchical adaptive loco-manipulation control for quadruped robots," *arXiv preprint arXiv:2209.13145*, 2022.

[10] H. Ferrolho, V. Ivan, W. Merkt, I. Havoutis, and S. Vijayakumar, "Roloma: Robust loco-manipulation for quadruped robots with arms," *arXiv preprint arXiv:2203.01446*, 2022.

[11] M. Mittal, D. Hoeller, F. Farshidian, M. Hutter, and A. Garg, "Articulated object interaction in unknown scenes with whole-body mobile manipulation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1647–1654.

[12] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter, "Circus anymal: A quadruped learning dexterous manipulation with its limbs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2316–2323.

[13] Y. Ji, G. B. Margolis, and P. Agrawal, "Dribblebot: Dynamic legged manipulation in the wild," *arXiv preprint arXiv:2304.01159*, 2023.

[14] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion," in *Conference on Robot Learning*. PMLR, 2023, pp. 138–149.

[15] K. N. Kumar, I. Essa, and S. Ha, "Cascaded compositional residual learning for complex interactive behaviors," *arXiv preprint arXiv:2212.08954*, 2022.

[16] X. Cheng, A. Kumar, and D. Pathak, "Legs as manipulator: Pushing quadrupedal agility beyond locomotion," *arXiv preprint arXiv:2303.11330*, 2023.

[17] C. Wang, S. Wang, B. Romero, F. Veiga, and E. Adelson, "Swingbot: Learning physical features from in-hand tactile exploration for dynamic swing-up manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5633–5640.

[18] A. Kloss, M. Bauza, J. Wu, J. B. Tenenbaum, A. Rodriguez, and J. Bohg, "Accurate vision-based manipulation through contact reasoning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6738–6744.

[19] C. Song and A. Boularias, "A probabilistic model for planar sliding of objects with unknown material properties: Identification and robust planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5311–5318.

[20] N. Mavrakis, R. Stolkin *et al.*, "Estimating an object's inertial parameters by robotic pushing: a data-driven approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9537–9544.

[21] Z. Xu, J. Wu, A. Zeng, J. B. Tenenbaum, and S. Song, "Densephysnet: Learning dense physical object representations via multi-step dynamic interactions," *arXiv preprint arXiv:1906.03853*, 2019.

[22] K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik, "Learning visual predictive models of physics for playing billiards," *arXiv preprint arXiv:1511.07404*, 2015.

[23] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv preprint arXiv:2107.04034*, 2021.

[24] G. Ji, J. Mun, H. Kim, and J. Hwangbo, "Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.

[25] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5078–5084.

[26] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 149–12 155.

[27] A. Petrenko, A. Allshire, G. State, A. Handa, and V. Makoviychuk, "Dexpbt: Scaling up dexterous manipulation for hand-arm systems with population based training," *arXiv preprint arXiv:2305.12127*, 2023.

[28] A. Allshire, M. MittaI, V. Lodaya, V. Makoviychuk, D. Makoviichuk, F. Widmaier, M. Wüthrich, S. Bauer, A. Handa, and A. Garg, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 802–11 809.

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

[30] J. Hwangbo, J. Lee, and M. Hutter, "Per-contact iteration method for solving contact dynamics," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018. [Online]. Available: www.raisim.com