HOMIE: Humanoid Loco-Manipulation with Isomorphic Exoskeleton Cockpit

Qingwei Ben[1,2,*], Feiyu Jia[1,*], Jia Zeng[1], Junting Dong[1], Dahua Lin[1,2], Jiangmiao Pang[1]
[1] Shanghai Artificial Intelligence Laboratory
[2] Multimedia Laboratory, The Chinese University of Hong Kong
[*] Authors with equal contribution

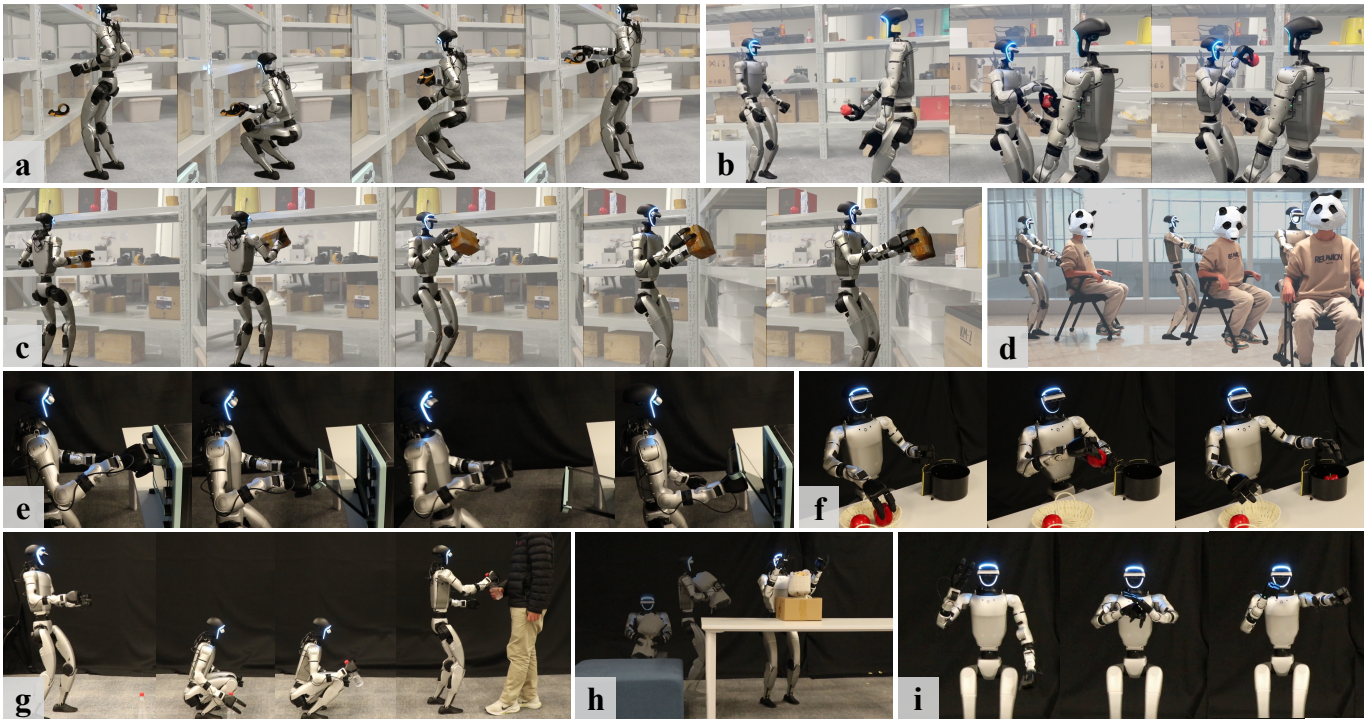


Fig. 1: HOMIE empowers the humanoid robot to execute various loco-manipulation tasks in the real world. (a): Squatting to grasp a tape and placing it on a higher shelf; (b): Facilitating an apple handover between two robots; (c): Holding a box and transferring it to another shelf; (d): Pushing a person seated on a chair; (e): Opening an oven door; (f): Picking up a tomato, handing it over, and placing it in a fruit basket; (g): Retrieving a bottle from the ground; (h): Holding a flower and placing it into a box on a table. (i): Balancing under vigorous motion. These tasks show robustness and generality of HOMIE.

***Abstract*—Current humanoid teleoperation systems either lack reliable low-level control policies, or struggle to acquire accurate whole-body control commands, making it difficult to teleoperate humanoids for loco-manipulation tasks. To solve these issues, we propose HOMIE, a novel humanoid teleoperation cockpit integrates a humanoid loco-manipulation policy and a low-cost exoskeleton-based hardware system. The policy enables humanoid robots to walk and squat to specific heights while accommodating arbitrary upper-body poses. This is achieved through our novel reinforcement learning-based training framework that incorporates upper-body pose curriculum, height-tracking reward, and symmetry utilization, without relying on any motion priors. Complementing the policy, the hardware system integrates isomorphic exoskeleton arms, a pair of motion-sensing gloves, and a pedal, allowing a single operator to achieve full control of the humanoid robot. Our experiments show our cockpit facilitates more stable, rapid, and precise humanoid loco-manipulation teleoperation, accelerating task completion and eliminating retargeting errors compared to inverse kinematics-based methods. We also validate the effectiveness of the data collected by our cockpit for imitation learning. Our project is fully open-sourced, demos and code can be found in our website.**

I. Introduction

Humans consistently rely on their physical agility to walk, squat to specific heights, and manipulate objects when performing tasks such as transferring goods in warehouses. While humanoid robots are expected to eventually take over these labor-intensive tasks, achieving full autonomy remains a significant challenge. In the interim, a robust teleoperation system is essential, enabling operators to seamlessly control robots for complex loco-manipulation tasks. These systems not only allow humans to guide robots with precision and fluidity but also serve as a critical tool for collecting high-

quality demonstrations to advance autonomous capabilities. Two indispensable components are required to realize this vision: (1) a robust loco-manipulation policy that enables robots to extend operational workspace; (2) a method that allows a single operator to precisely control the robot's upper-body manipulation and lower-body locomotion simultaneously.

However, most existing efforts to teleoperate humanoid robots focus solely on controlling the upper body [1, 2, 3, 4, 5]. This restriction significantly limits the robot's operational workspace, as it prevents the robot from moving freely or adjusting its posture (e.g., squatting) to adapt to different task requirements. While some researchers have explored equipping humanoids with loco-manipulation capabilities through reinforcement learning (RL) [6, 7, 8, 9, 10, 11], these approaches often fail to achieve the necessary range of motion, such as squatting to specific heights, which is critical for many real-world tasks. Additionally, many humanoid whole-body teleoperation methods require operators to physically move to control the robot's motion, making it challenging to maintain precise and stable full-body control over extended distances. Mainstream teleoperation systems typically rely on vision-based methods [1, 3, 4, 5] or heterogeneous exoskeletons [2] to determine end-effector poses, followed by inverse kinematics (IK) to compute joint positions. Nonetheless, both pose estimation and IK solving introduce inaccuracies, which compromises the precision and reliability of teleoperation.

To address these issues, we introduce HOMIE, a novel humanoid teleoperation cockpit composed of a humanoid loco-manipulation policy and an exoskeleton-based hardware system. This cockpit enables a single operator to precisely and efficiently control a humanoid robot's full-body movements for diverse loco-manipulation tasks. Integrated into simulation environments, our cockpit also enables seamless teleoperation in virtual settings. Specifically, we introduce three core techniques to our RL-based training framework: upper-body pose curriculum, height tracking reward, and symmetry utilization. These components collectively enhance the robot's physical agility, enabling robust walking, rapid squatting to any required heights, and stable balance maintenance during dynamic upper-body movements, thereby significantly expanding the robot's operational workspace beyond existing solutions. Unlike previous whole-body control methods that depend on motion priors derived from motion capture (MoCap) data [12], our framework eliminates this dependency, resulting in a more efficient pipeline. Our hardware system features isomorphic exoskeleton arms, a pair of motion-sensing gloves, and a pedal. The pedal design for locomotion command acquisition liberates the operator's upper body, enabling simultaneous acquisition of upper-body poses. Since the exoskeleton arms are isomorphic to the controlled robot and each glove has 15 degrees of freedom (DoF), which is more than most existing dexterous hands, we can directly set upper-body joint positions from the exoskeleton readings, dispensing with IK and achieving faster and more accurate teleoperation. Moreover, our gloves can be detached from the arms, allowing them to be reused in systems isomorphic to different robots. The total cost of the hardware system is only \$0.5k, which is significantly lower than that of MoCap devices [13].

Through ablation experiments, we validate the effectiveness of each technique in our training framework and demonstrate the robustness of the resulting policies across different robots. Our evaluation shows that the hardware system supports 200% faster and more accurate pose acquisition than previous methods, enabling operators to complete tasks more efficiently than virtual reality (VR)-based approaches. Real-world studies confirm that the trained policies can be deployed directly in the real world, allowing robots to perform diverse loco-manipulation tasks stably in complex environments. We further show that real-world data collected via HOMIE can be effectively used by imitation learning (IL) algorithms, allowing humanoid robots to autonomously execute tasks.

In summary, our core contributions are three-fold:

1) We propose HOMIE, the first implementation of a cockpit for humanoid whole-body loco-manipulation teleoperation enables a single operator to fully control the robot and execute diverse tasks seamlessly.
2) We first achieve robust humanoid loco-manipulation including squatting under arbitrary continuous changing upper-body poses without using any motion prior.
3) Our hardware system supports more precise and rapid whole-body control of humanoids than previous works, reducing task completion times by nearly half.

II. Related Works

A. Dual Arm Robot Teleoperation

Teleoperating dual-arm robots to perform complex manipulation tasks is an efficient way to collect real-world expert demonstration, which can then be used by IL to learn autonomous skills [14, 1, 3, 17, 18]. Some researchers utilize robotic arms identical to the teleoperated ones [19, 14, 15, 16], making joint-matching possible, thus ensuring high accuracy and fast response speed. However, due to the high cost of robotic arms, the establishment of such a system incurs significant expenses. Additionally, teleoperating dexterous hands with these systems is not feasible. An alternative approach is to use VR devices [1, 5] or just a camera [20, 4, 21]. These works use vision-based techniques to capture the operator's wrist postures and key points of the hands, which are used by IK to calculate the joint positions of the arms and hands. However, due to limitations in the accuracy, inference speed, and difficulty in handling occlusions of pose estimation, such approaches cannot guarantee rapid and accurate pose acquisition. Some researchers try to use MoCap methods [13, 22, 23, 24] to acquire more accurate poses at higher frequencies, but MoCap equipment is very expensive. Moreover, since IK is an iterative method that approximates solutions, even when wrist and hand poses are captured accurately, the limitations of IK may prevent the robot from achieving the desired posture. Another possible solution is an exoskeleton-based teleoperation system, which does not require an additional identical robot, thus the overall cost is relatively low. Some research calculates the end-effector pose of the exoskeleton using Forward Kinematics

TABLE I: **Comparison between representative teleoperation systems and HOMIE.** Cost: total cost of each system. Arm and Dex-Hand Tracking: method of tracking arm and hand poses. Loco-Manip.: whether or not have loco-manipulation capability. Whole-body: whether or not teleoperate the whole body of humanoid robots. No Mocap: whether or not exclude MoCap data.

Teleop System	Cost ($)	Arm Tracking	Dex-Hand Tracking	Loco-Manip.	Whole-body	No MoCap
Mobile-ALOHA [14]	32k	Joint-matching	✗	✓	✗	✓
GELLO [15]	0.6k	Joint-matching	✗	✗	✗	✓
AirExo [16]	0.6k	Joint-matching	✗	✗	✗	✓
ACE [2]	0.6k	Joint-matching	Vision Retarget	✗	✗	✓
DexCap [13]	4k	Vision Retarget	Mocap + SLAM	✗	✗	✓
AnyTeleop [4]	$\sim$ 0.3k	Vision Retarget	Vision Retarget	✗	✗	✓
OpenTelevision [1]	4k	VR devices	VR devices	✗	✗	✓
HumanPlus [6]	0.05k	Vision Retarget	Vision Retarget	✓	✓	✗
OmniH2O [8]	0–3.5k	Vision / VR	Vision / VR	✓	✓	✗
Mobile-TeleVision [11]	3.5k	VR devices	VR devices	✓	✓	✗
HOMIE (Ours)	0.5k	Joint-matching	Joint-matching	✓	✓	✓

(FK) and then apply IK to determine the robot's joint positions, while using computer vision techniques to capture the hand poses [2]. However, these systems are also limited by the inaccuracies of IK and pose estimation. Some studies utilize isomorphic exoskeletons [15, 16], which can also employ joint-matching to teleoperate the robots, ensuring both low cost and high accuracy and control frequency. Neverthless, these systems typically handle robotic arms equipped with grippers, limiting their application to basic manipulation tasks rather than dexterous ones. Since some projects have introduced cheap and reliable motion-sensing gloves [25, 26], redesigning and combining them with an exoskeleton could potentially overcome this limitation, a solution that has not yet been realized in this field. HOMIE is designed to combine all the advantages mentioned above, integrating isomorphic exoskeleton arms with a pair of novel motion-sensing gloves. We will introduce this system in Sec. III-C. A comparison between HOMIE and previous representative teleoperation systems can be found in Tab. I.

B. Humanoid Whole-body Loco-Manipualtion

To enable robots to perform whole-body loco-manipulation tasks, some researchers focus on model-based optimization algorithms [27, 28, 29, 30, 31], particularly generating locomotion control laws by solving optimal control problems (OCPs). Despite significant efforts to make OCPs computationally tractable, these algorithms still struggle with complex scenarios due to their high computational demands during online processing. Reinforcement Learning (RL)-based algorithms, especially those based on Proximal Policy Optimization (PPO) [32], offer a more powerful alternative. Using these methods, several studies successfully achieve whole-body loco-manipulation in quadruped robots [33, 34, 35, 36], and some teach humanoid robots to traverse various terrains [37, 38, 39, 40, 41, 42, 43, 44] or perform parkour [45]. These achievements motivate researchers to apply the same techniques to humanoid whole-body loco-manipulation [46]. Some studies train whole-body policies for humanoid robots [10], enabling them to act in a manner similar to human operators or even dance with people. Some other research separates the upper and lower body [9, 6, 7, 8, 11], using policies trained by RL to control the lower body while directly setting the joint positions of the upper body, thus helping robots achieve better balance. Despite achieving impressive results, these methods still face several common limitations. First, they often rely on retargeted MoCCap data [12] to get motion prior [47] for training robots. However, obtaining MoCap data is costly, and adapting robots to new poses necessitates additional data collection, which significantly hinders the scalability of these approaches. Second, many of these methods employ vision-based algorithms to estimate the operator's poses, which lack the precision of exoskeleton-based devices. This limitation reduces the accuracy required for humanoid robots to perform loco-manipulation tasks effectively. Third, these methods generally fail to incorporate the ability to control a robot's body height. Height control is crucial for handling objects at varying elevations, and its absence severely restricts the robot's operational workspace. Finally, when issuing locomotion commands, some studies rely on body movement data directly [9, 6, 7, 8], while others use joysticks or pedals [11]. The former approach becomes impractical when operators need to control robots in large-scale environments, whereas the latter offers a more effective solution. However, controlling with joysticks necessitates the use of hands, which may already be occupied by other manual tasks, thereby highlighting the advantage of using pedals for locomotion commands.

III. Method

A. System Overview

As shown in Fig. 2, HOMIE consists of a low-level policy π_{loco} and an exoskeleton-based hardware system. At any given time t, the operator inside the cockpit observes the first point of view (FPV) of the robot through the display. By stepping on the pedal, the operator provides the required locomotion

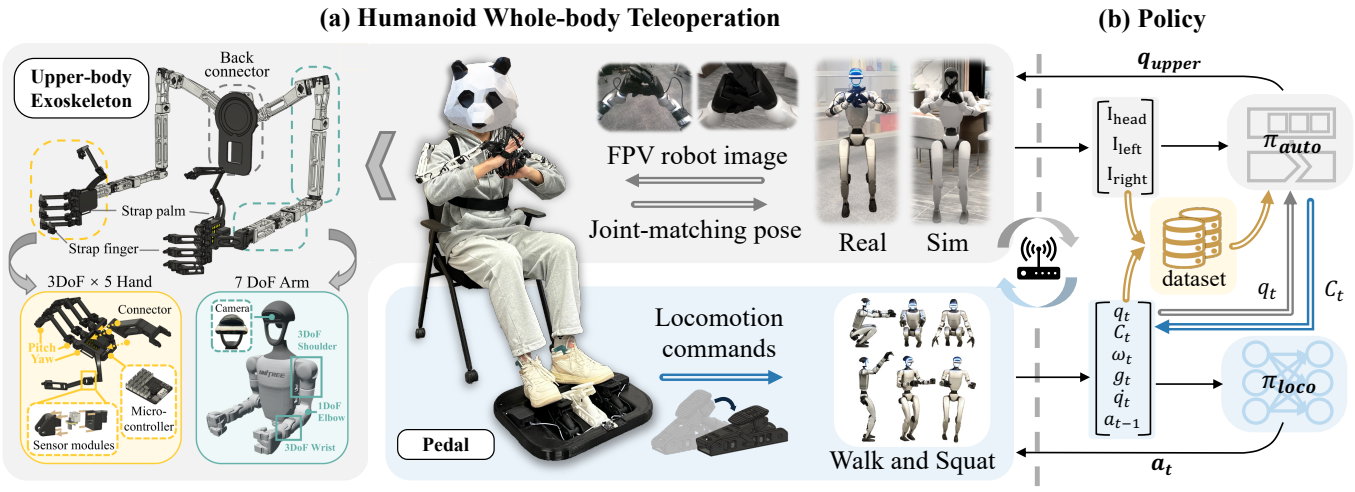


Fig. 2: **System Overview.** (a): how an operator uses the exoskeleton-based hardware system to control humanoid robots in the real world and simulation. (b): how π_{loco} controls the robots, the data collection process for training π_{auto}, and how π_{auto} takes over the operator to control the robots. Communication between the cockpit and the robot is achieved via Wi-Fi.

commands $C_t = [v_{x,t}, \omega_{yaw,t}, h_t]$ where $v_{x,t}$ is the desired forward or backward speed, $\omega_{yaw,t}$ is the turning speed, and h_t is the target height of the robot's torso. The policy π_{loco} controls the robot's lower-body based on C_t. Meanwhile, the operator controls the exoskeleton to provide the required joint angles q_{upper} for the robot's upper-body, which are directly set to the robot. The upper and lower bodies work in coordination, continuously cycling through the process, ultimately enabling teleoperating robots to complete loco-manipulation tasks either in the real world or in simulation. Communications between the cockpit and the robot are achieved via Wi-Fi, allowing operation even when the robot is far from the hardware system. We can collect demonstrations while teleoperating the robot and use them to train an autonomous policy π_{auto}. Once trained successfully, π_{auto} can take over the operator to give C_t and q_{upper}, thus driving the robot to perform tasks autonomously.

B. Locomotion Policy

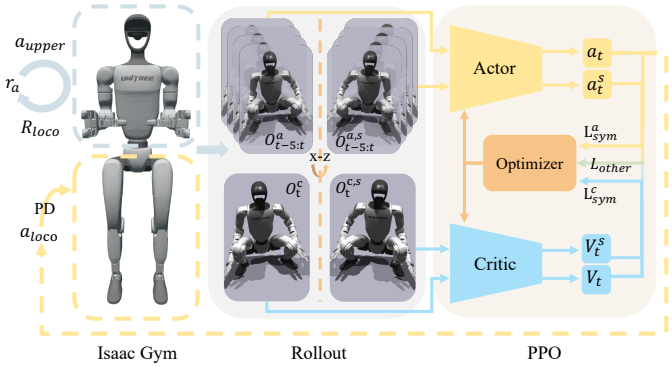


Fig. 3: RL training framework of HOMIE.

To enable humanoid robots to perform loco-manipulation tasks, we design an RL-based training framework, which trains different robots to accomplish squatting and walking under continuously changing upper-body poses. We take Unitree G1 as an example and show the process of the framework in Fig. 3. The policy π_{loco} trained by this process is capable of zero-shot sim-to-real transfer. We introduce the training settings and three key techniques of our framework in this section.

1) Training Settings: The observations of one step are defined as $O_t = [C_t, \omega_t, g_t, q_t, \dot{q}_t, a_{t-1}]$, where C_t is the command, ω_t is the body's angular velocity, g_t is the projection of $\vec{g} = [0, 0, -1]$ in the robot's torso coordinate frame, q_t is the joint angles of all joints of the robot, $\dot{q}_t$ is the joint velocities of all joints of robot, a_{t-1} is the last time action. Then we can get the whole observations of π_{loco} by concatenating $O_{t-5:t}$. The actions a_t of the policy correspond one-to-one with the joints of the robot's lower body. After the neural network computes a_t based on $O_{t-5:t}$, we use

$$\tau_{t,i} = Kp_i \times (a_{t,i} - q_{0,t,i}) - Kd_i \times \dot{q_{t,i}} \quad (1)$$

to calculate the torques for joint motors, thereby driving the motors to work and enabling the robot's movement. In the equation, i denotes the index of joints, $\{Kp_i\}$ and $\{Kd_i\}$ are stiffness and damping of each joint, $\{q_{0,t,i}\}$ are default joint positions of each joint. Our framework is implemented based on the code of [48, 49], and more training details can be found in Appendix A.

2) Upper-body Pose Curriculum: We use a curriculum learning technique to ensure that π_{loco} can still complete locomotion tasks under any continuously varying poses of the robot's upper-body. We adjust the sampling range of the upper body joint angles using the upper action ratio r_a. At the start of training, r_a is set to 0. Each time the policy drives the robot to track the linear velocity with a reward function that reaches the threshold, r_a increases by 0.05, eventually reaching 1. We

first sample r'_a from the probability distribution

$$p(x|r_a) = \frac{20(1-r_a)\, e^{-20(1-r_a)x}}{1-e^{-20(1-r_a)}}, \quad r_a \in [0,1) \tag{2}$$

and then resample a_i by $\mathcal{U}(0, r'_a)$. We actually sample a_i by

$$a_i = \mathcal{U}(0, -\frac{1}{20(1-r_a)} \ln\left(1 - \mathcal{U}(0,1)\left(1 - e^{-20(1-r_a)}\right)\right)) \tag{3}$$

As r_a increases, the probability distribution gradually transitions from being close to 0 to $\mathcal{U}(0,1)$. This ensures that during the curriculum process, the probability distribution consistently satisfies $p(x|r_a) > 0, \forall x \in [0,1], r_a \in [0,1)$. Compared to directly using $\mathcal{U}(0, r_a)$, this method approaches the final target in a more gradual and smoother manner. For better understanding of Eq. (3), we visualize it in Appendix A. To simulate the continuous changes in upper body movements when controlled by our cockpit, we resample target upper-body poses every 1 second according to the above process. We then use uniform interpolation to ensure that the target movement gradually changes from the current value to the desired value over the 1-second interval. Without this approach, we find that the robot struggles to maintain balance under continuous motions.

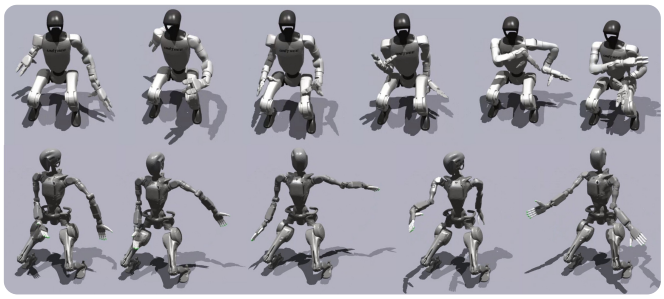


Fig. 4: Different robots are trained to walk and squat with continuous changing upper-body poses in Isaac Gym.

3) Height Tracking Reward: Tracking heights can significantly expands the feasible operational workspace of humanoid robots, thus helping the robots perform more loco-manipulation tasks. Therefore, π_{loco} needs to enable the robot to squat to the target height h_t. To achieve this, we design a new reward function

$$r_{knee} = -\|(h_{r,t} - h_t) \times (\frac{q_{knee,t} - q_{knee,min}}{q_{knee,max} - q_{knee,min}} - \frac{1}{2})\|, \tag{4}$$

where $h_{r,t}$ is the robot's actual height, $q_{knee,min}$ and $q_{knee,max}$ are the maximum and minimum actions of knee joints, $q_{knee,t}$ is the current positions of robot's knee joints. r_{knee} encourages flexion of the knee joints when $h_{r,t} < h_t$, and encourages extension when $h_{r,t} > h_t$. In the training process, we resample all commands every 4 seconds. At this point, one-third of the environments are randomly selected to train the robot to squat, while the remaining two-thirds focus on teaching the robot to stand and walk. This strategy helps balance the learning of squatting and walking. Additionally, the same environment switches between learning to squat and learning to walk, enabling the policy to smoothly transition between squatting and walking tasks. For better understanding of Eq. (4), we visualize it in Appendix A.

4) Symmetry Utilization: We introduce the same trick as [50] to our training framework. Each time we obtain a transition $T_t = (s_t, a_t, r_t, s_{t+1})$ from the simulation, we perform a flip operation on it. Specifically, we apply symmetry to the actor and critic observations with respect to the robot's x-z plane. This involves flipping elements such as the positions, velocities, and actions of the robot's left and right joints, as well as the desired turning velocity, across the x-z plane to obtain a mirrored transition T'_t. Both T_t and T'_t are then added to the rollout storage. This process helps to improve data efficiency and ensure symmetry in the sampled data, reducing the likelihood of the trained policy being asymmetrical in terms of left and right performance. In the learning phase, we also apply this procedure to the samples T_t got from the rollout storage to get T'_t. Both T_t and T'_t are passed through the actor and critic networks to obtain a_t, a'_t, V_t, V'_t respectively, which are used to calculate additional losses:

$$\mathcal{L}_{sym}^{actor} = MSE(a_t, a'_t), \tag{5}$$

$$\mathcal{L}_{sym}^{critic} = MSE(V_t, V'_t). \tag{6}$$

These two losses are added to the network optimization process, thereby enforcing symmetry of the neural network.

C. Hardware System Design

To enable a single operator to control the full body of humanoid robots, we design a low-cost exoskeleton-based hardware system as shown in the left part of Fig. 2. For the upper-body teleoperation of the humanoid robots, we design 3D-printed 7-DoF isomorphic exoskeleton arms for precise mapping of the upper limb joint angles, specifically tailored for two types of humanoid robots: Unitree G1 and Fourier GR-1. Additionally, we design a pair of low-cost motion-sensing gloves capable of mapping up to to 15 DoF of finger angles. For locomotion command acquisition, we design a foot pedal that simulates the press-and-release actions of the foot during driving, enabling control of the humanoid robot's movements such as walking and squatting. The operator can easily deploy this system and perform single-person teleoperation of the robot's loco-manipulation, similar to driving a car in a cockpit or playing a racing game.

*1) **Isomorphic Exoskeleton**:* To achieve accurate control and mapping of the upper limb joints of the humanoid robots, we employ an isomorphic exoskeleton as the teleoperation solution for controlling the robot's upper body. Based on the morphology of the Unitree G1 and Fourier GR-1, our isomorphic exoskeleton design consists of a symmetric pair of arms, each with 7 DoF, corresponding to the 7 DoF of each arm of the robot (3 DoF for the shoulder, 1 DoF for the elbow, and 3 DoF for the wrist). Each joint of the exoskeleton is equipped with a DYNAMIXEL XL330-M288-T servo, which provides joint angle readings and adjustments with an accuracy of 0.09°, enabling precise joint angle mapping and initial

calibration. The exoskeleton's operational part is designed to match the length of the human arms. Considering the challenge of fully replicating the robot's upper arm structure, we align the servos with the robot's motor URDF joint coordinate system. By proportionally mapping the relative positions of the coordinate axes to the relative positions of the servos, we ensure that both the operator's experience and the system's isomorphic nature are preserved. Our isomorphic exoskeleton is fixed to the operator's back using adjustable straps. This setup minimizes interfere with the operator's basic movements while covering most of the upper-body motion range of the humanoid robot.

*2) **Motion-sensing Gloves**:* For fine teleoperation of the fingers, we adopt a joint-matching approach. Based on the Nepyone glove project [25], we design a low-cost motion-sensing glove that connects directly to the exoskeleton for assembly and use, providing up to 15 DoF for finger capture to control dexterous hands. Specifically, each finger is equipped with three sets of sensors, which map the pitch motion of the finger tip and finger pad, as well as the yaw motion of the finger pad. This setup is sufficient to enable the mapping of different dexterous hands for humanoid robots. We place Hall effect sensors and small neodymium magnets at each joint. When the joint rotates, the neodymium magnet rotates as well, thereby affecting the magnetic field sensed by the sensor and achieving the mapping of finger joint angles. Additionally, we design the glove's microcontroller, sensor modules, and structural model. The microcontroller is mounted on the back of the hand and can be directly connected to the packaged sensors using terminal connectors, allowing for easy plugging and unplugging to reassign and modify the mapping relationships. Our motion-sensing gloves can be easily attached to and detached from different exoskeletons, offering high versatility.

*3) **Foot Pedal**:* In our cockpit, the foot pedal is used as a replacement for a remote controller, enabling command control of the humanoid robot's lower body by giving commands C_t to π_{loco}. The operator controls the acceleration and deceleration of the robot's lower body movement by pressing and releasing the foot pedal. We use high-precision rotary potentiometers to map pedal pressure changes to electrical signals. In our system, we control the humanoid robot's locomotion through linear velocity, yaw velocity, and height adjustment. These commands allow the robot to fully demonstrate its locomotion capabilities. To achieve this, we use three small pedals to control these commands. Additionally, a pair of mode-switching buttons (foot-operated with momentary switches) are used to toggle between forward/backward and left/right turning directions, as shown in Fig. 5. Users can modify the pedal configuration and reassign commands to adapt to diverse movement combinations.

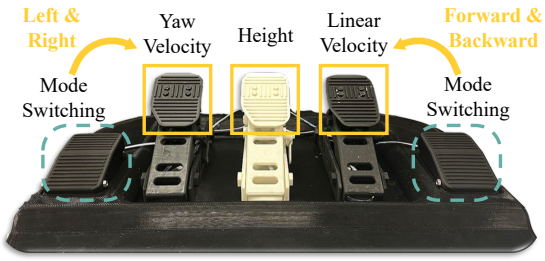


Fig. 5: **Pedal command control.** The three small pedals respectively control $[0, \pm\omega_{max}]$, $[H_{min}, H_{max}]$, and $[0, \pm V_{max}]$. The left-side switching button is used to toggle between left and right modes, while the right-side switching button is used to toggle between forward and backward modes.

IV. Experiment

A. Locomotion Policy

1) Ablation of training framework: In this section, we perform ablation experiments on the proposed upper-body pose curriculum, the height tracking reward, and the use of symmetry. All ablation experiments are conducted based on the methods described in Sec. III-B. For each setting, we use three random seeds to train policies for Unitree G1 and evaluate them in 1000 environments over a 20-second evaluation period with random upper-body poses sampled from Eq. (3) with $r_a \to 1$. Metrics for evaluation are tracking linear velocity error, tracking angular velocity error, tracking height error, symmetry loss and living time. The final performance for each setting is obtained by computing the average and standard deviation of the results across the three policies trained from three random seeds. All trainings are conducted on Nvidia RTX 4090 and simulated by Isaac Gym with 4096 parallel environments, where components unrelated to the ablation are kept unchanged, and only relevant parts are modified for training. Detailed parameters used in training and evaluation processes are listed in Appendix A. We mark the setting of our proposed method as **ours** in the following sections.

Upper-body Pose Curriculum. We compare **ours** against two alternatives: **w/o cur**, which omits the curriculum and directly samples $a_i = \mathcal{U}(0, \mathcal{U}(0, 1))$, and **rand**, which uses the same r_a curriculum but replaces Eq. (3) with $a_i = \mathcal{U}(0, \mathcal{U}(0, r_a))$. Since all three methods adopt the same sampling strategy $a_i = \mathcal{U}(0, \mathcal{U}(0, 1))$ as $r_a \to 1$, the final objective remains consistent, ensuring a fair comparison. The experimental results, shown in the first row of Fig. 6, reveal that **ours** outperforms both **w/o cur** and **rand** in linear velocity tracking, angular velocity tracking, and height error, with faster convergence and smaller errors. There is no significant difference between **w/o cur** and **rand** in the final results for these metrics. Given that the symmetry loss can reach values on the order of 20 without constraints, no significant difference is observed across the three methods in terms of symmetry loss. All three configurations achieve similar final living times, but **ours** and **w/o cur** converge more quickly. **Rand**, despite employing some curriculum adjustments, is limited by r_a, and values in the range $(r_a, 1]$ are not sampled during training, making it harder for the model to converge as r_a increases. In contrast, both **ours** and **w/o cur** sample the full $[0, 1]$ range from the beginning, enabling faster and more stable convergence. Thus, our curriculum approach leads to better

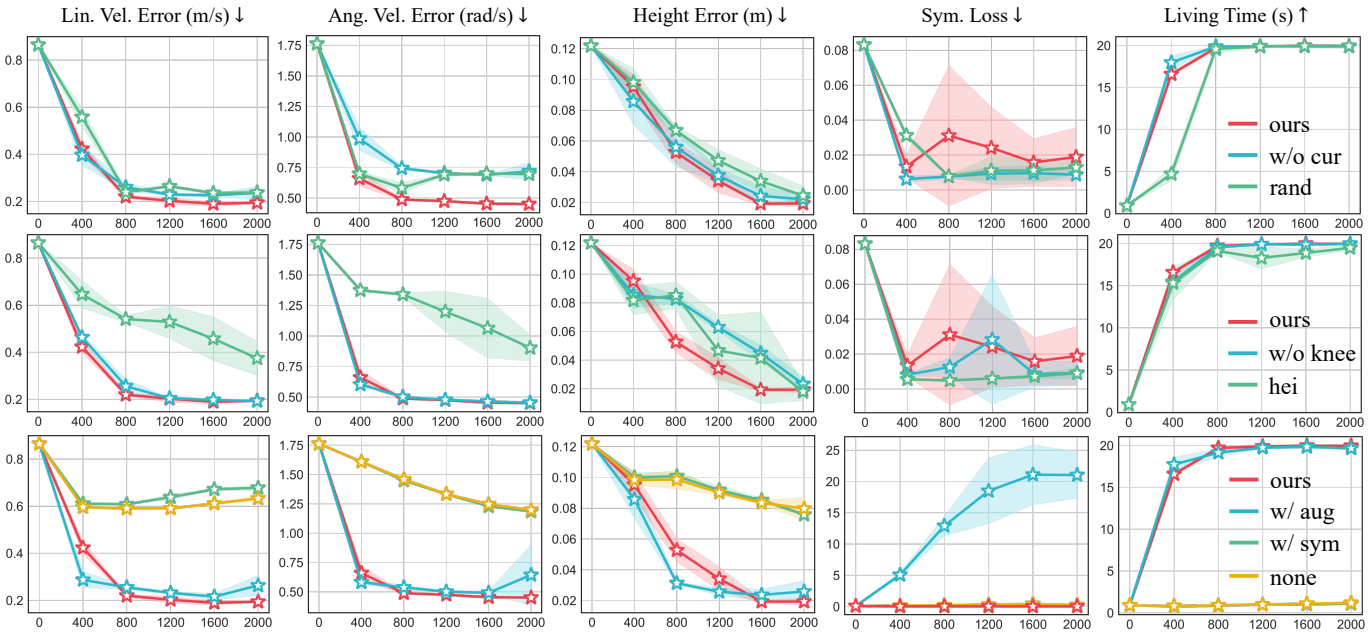


Fig. 6: **Ablation experiments of our RL training framework.** Each row from top to bottom represents the ablation study for upper-body curriculum, height tracking reward, and symmetry utilization, respectively. Each column represents the evaluation of the corresponding metrics for checkpoints under different ablation settings. The ↑ and ↓ symbols beside the metrics indicate whether a higher or lower value is better for the respective metric.

performance compared to **rand**. Although **w/o cur** does not use a curriculum, allowing a_i to continuously sample from $[0, 1]$, the lack of difficulty smoothing leads to worse final tracking results, highlighting that our curriculum design offers a more effective training process.

Height Tracking Reward. We design two additional algorithms **w/o knee**, which does not use r_{knee} described in Eq. (4) and **hei**, which also omits r_{knee} but increases the scale of the height tracking reward. We show the results in the second row of Fig. 6. As shown in the figure, none of the three settings cause significant changes in the symmetry loss during training. In terms of linear velocity error and angular velocity error, **ours** and **w/o knee** perform similarly, while **hei** shows much larger errors. For height error, our method converges faster than both **w/o knee** and **hei**, even though **hei** initially performs better (at 400 steps). There is no significant difference among the three settings in terms of living time. These results indicate that just scaling up the height tracking reward in **hei** may initially lead to faster reduction in height tracking error, but it negatively affects the feedback from other rewards, preventing the robot from balancing multiple tasks effectively. In fact, **hei** ultimately does not achieve faster convergence in height tracking compared to **ours**. In contrast, the inclusion of **rknee** in our method provides more specific guidance for squat tracking, allowing the robot to reduce tracking error and converge more quickly. This highlights the effectiveness of **rknee** in helping the robot learn squat motions.

Symmetry Utilization. We introduce three algorithmic variants for comparison with **ours** in terms of symmetry utilization: **w/ aug**, which uses only symmetrical data augmentation; **w/ sym**, which only uses symmetry loss; and **none**, which does not employ symmetrical data augmentation or symmetry loss. Testing results are presented in the third row of Fig. 6. Except for symmetry loss, the performance of **ours** and **w/ aug** is similar. However, when considering overall tracking accuracy, ours performs slightly better. On the other hand, **w/ aug** exhibits a very high symmetry loss, suggesting that using symmetry loss helps maintain the robot's left-right symmetry in the learned policy. This indirectly supports the idea that a symmetric policy benefits the robot's locomotion tasks [50]. Both **nsym** and **none** show a tendency for improvement, but their training speed is much slower. Notably, a direct comparison between **w/ sym** and **none** reveals that **w/ sym** achieves lower symmetry loss. However, due to slower training, **none** exhibits less symmetry breaking compared to **w/ aug**. In summary, symmetry data augmentation significantly improves training efficiency, while the use of symmetry loss effectively prevents the policy from sacrificing symmetry to complete tasks and also benefits the task itself.

2) Training on Different Robots: We select another kind of robot, Fourier GR-1, which is quite different from Unitree G1, to demonstrate the generality of our approach across different robot models. As shown in Fig. 7, Fourier GR-1 is much taller and heavier than Unitree G1 while having lower hand weight ratio. Compared to the training setting of Unitree G1, we only change the range of height tracking and some robot-specific distance values, without any other changes in reward scales or training pipeline. We evaluate the policy trained after 2k steps of each robot with metrics used in Sec. IV-A1 and present them in Tab. II. The results demonstrate that even though these two

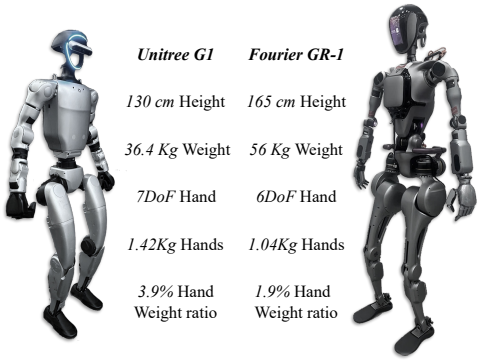


Fig. 7: Key parameters of Unitree G1 (left) and Fourier GR-1 (right). Hand weight ratio = total weight / hands weight.

kinds of robots are quite different, our RL training framework can train them to converge to a policy which can drive robots to perform locomotion and squatting tasks robustly under any upper-body poses. Training details for Fourier GR-1 can be found in Appendix A.

TABLE II: Evaluation of different robots trained with our RL training framework

Metrics	Unitree G1	Fourier GR-1
Lin. Vel Error (m/s)	0.194 (±0.003)	0.273 (±0.003)
Ang. Vel Error (rad/s)	0.451 (±0.006)	0.540 (±0.002)
Height Error (m)	0.022 (±0.019)	0.038 (±0.003)
symmetry loss (-)	0.019 (±0.017)	0.009 (±0.001)
Living Time (s)	19.947 (±0.092)	19.960 (±0.035)

B. Teleoperation Hardware Performance

TABLE III: Hardware Indicators of three component of the hardware system.(Freq.:frequency, Acc.:accuracy)

Hardware	Cost ($)	Acquisition Freq.	Acquisition Acc.
Exoskeleton	430	0.26 kHz	2^{12}(with 360°)
Glove	30 (each)	0.3 kHz	2^{12}
Pedal	20	0.5 kHz	2^{12}(with 270°)

We list a series of hardware indicators for our teleoperation hardware system consisting of isomorphic exoskeleton arms, a pair of motion-sensing exoskeleton gloves, and a pedal in Tab. III. We detail their costs, with the primary expense attributed to the exoskeleton section. This is because we independently design and solder the control boards (PCBs) and sensor modules for the motion-sensing gloves and the pedal components. The acquisition frequency represents the update signal frequency measured between the hardware components of the teleoperation system and the host computer via a wired connection at a baud rate of 115200. Changing the baud rate can affect the acquisition frequency. The acquisition accuracy represents the range of angular change (in degrees) and the corresponding variation in acquisition readings, ranging from 0 to 4095(2^{12}). Since the mapping relationship for the motion-sensing gloves is not a clearly defined linear one, and the mapping angles for each finger joint vary, more detailed information can be found in the Appendix B.

For upper-body teleoperation, the task can be divided into two parts: arm control and dexterous hand control. We select the arm pose frequency and hand pose frequency as evaluation metrics, which measures the smoothness and fluidity of teleoperation. In Tab. IV, we compare the visual and VR schemes with our joint-matching scheme. Since our joint-matching scheme directly sets the robot's upper-body poses without the need for additional time-consuming processes, the output frequency to the robot closely matches the acquisition frequency. Therefore, our approach achieves a very high output frequency without requiring GPU and System on Chip (SoC) intensive hardware.

TABLE IV: Upper-body teleoperation frequency of output to the robot's arm and hand. (SoC: System on Chip)

Teleop system	Hardware	Arm (Hz)	Hand (Hz)
Telekinesis [20]	2 RTX 3080 Ti	16	24
AnyTeleop [4]	RTX 3090	125	111
OpenTeleVision [1]	M2 Chip	60	60
Ours	**No GPU / SoC**	**263**	**293**

To further demonstrate the extensibility of our motion-sensing gloves, we test different types of dexterous hands from the Dex Retargeting library in AnyTeleop [4] within the SAPIEN [51] environment. The results are presented in Fig. 8, with the upper line shows names of tested dexterous hands while the lower line indicates number of joints of each hand.

Fig. 8: Controlling different types of dexterous hands in simulation with our motion-sensing gloves.

C. Teleoperation System

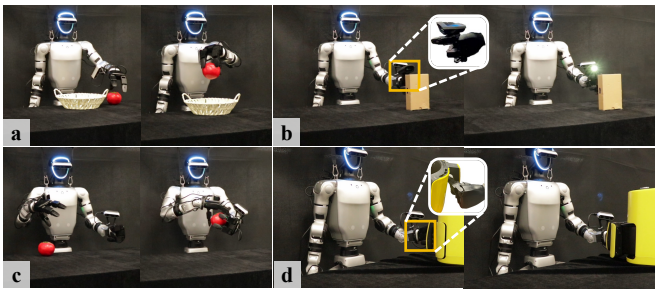


Fig. 9: Desktop tasks for comparison of completion time. a: Pick & Place; b: Scan Barcode; c: Hand Over; d: Open Oven.

1) Real World: We deploy the trained policy on the Unitree G1 in the real world and teleoperate it to perform various loco-manipulation tasks using our isomorphic exoskeleton hardware system. The deployment code for G1 is derived from [52]. Fig. 1 (a) and Fig. 1 (c) demonstrate the robot's capability to squat, pick objects from lower shelves, and place them on higher ones, as well as to grasp and transfer boxes between shelves utilizing its locomotion abilities. Fig. 1 (b) highlights the extensibility of our system, enabling two operators to control separate robots and collaboratively perform tasks, such as transferring apples. In Fig. 1 (d), the robot is controlled to push a 60 kg person sitting in a chair, who weighs roughly twice as much as the robot, demonstrating the robustness of the loco-manipulation system. Fig. 1 (e) illustrates how the robot uses its loco-manipulation abilities to open an oven by grasping the handle and moving backward simultaneously. Fig. 1 (f) shows that our teleoperation system is capable of performing dual-hand collaborative tasks, such as one hand passing an object to the other. Fig. 1 (g) demonstrates the robot's ability to grasp objects from low ground, while Fig. 1 (h) shows the robot's capability to lift and place heavy items, such as a bundle of flowers, into a box using both arms. Fig. 1 (i) demonstrates how the robot maintains balance with different upper-body poses. In all these tasks, each robot is controlled by a single operator, and the communication between the robot and operator is facilitated via Wi-Fi, without restricting the robot's movement space. These tasks showcase the robustness of our loco-manipulation policy and HOMIE 's ability to teleopeate humanoids perform a wide range of complex tasks in various environments.

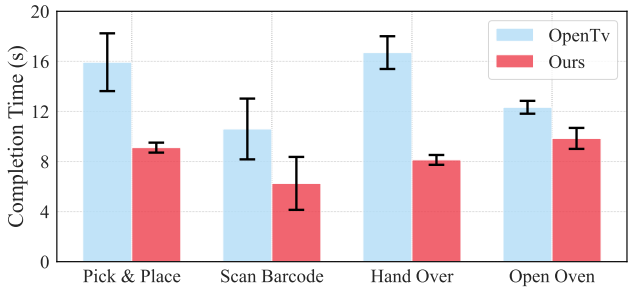


Fig. 10: Comparison of completion time to perform desktop tasks between our hardware system and OpenTelevision [1].

To demonstrate the efficiency of our teleoperation system, we compare the task completion time between our hardware system and a VR-based method, OpenTelevision [1], across four tasks as shown in Fig. 9. These tasks are designed to evaluate the system's ability to precisely control the robot's arms and hands in various scenarios: **Pick & Place:** The robot is required to grasp a tomato from the table and place it into a fruit basket. **Scan Barcode:** The robot must hold a scanner, press its button using a finger, and scan a barcode on a box. **Hand Over:** The robot needs to grasp a tomato and pass it to another hand. **Open Oven:** The robot must insert its finger into a handle and open the oven door. These tasks test key capabilities of teleoperation, including precise positioning, bimanual coordination, and fine-grained finger control. The results, shown in Fig. 10, indicate that our system achieves task completion times nearly half of those of the VR-based method. Notably, when tasks require precise positioning and orientation, the performance gap between our system and the VR method becomes even more pronounced. This is because VR-based pose estimation tends to perform poorly in tangential directions, whereas our exoskeleton-based approach avoids such issues entirely. These results demonstrate that our exoskeleton system enables operators to teleoperate robots more smoothly and efficiently, particularly in tasks requiring high precision and dexterity.

2) Simulation: We transfer the trained policies for Unitree G1 and Fourier GR-1 from Isaac Gym to a scene developed by GRUtopia [53], which is based on Isaac Sim and IsaacLab [54]. This migration enables the use of HOMIE to control robots within a variety of simulated environments. By leveraging these simulated scenes, the robots can perform diverse loco-manipulation tasks more cost-effectively and in a wider range of scenarios than would be feasible in the real world. As shown in Fig. 11, operators can seamlessly direct the robots' movements and actions in complex, realistic settings, demonstrating the versatility and applicability of HOMIE in diverse simulated contexts.

Fig. 11: **Simulation migration.** The upper row illustrates how the operator controls the robot with FPV to perform loco-manipulation tasks. The lower row demonstrates the robot navigating through realistic simulated environments.

D. Autonomous Policy

Fig. 12: Hardware Setup for Imitation Learning

1) Data Collection: To validate the effectiveness of the demonstratons collected by HOMIE for IL algorithms, we design two distinct tasks: **Squat Pick**: squatting to pick a tomato on the lower sofa; **Pick & Place**: picking and placing a tomato. We capture RGB images, robot states q_t, the upper-body commands q_{upper}, and the locomotion commands C_t at 10Hz, and collect 50 episodes per task. The hardware setup for image capture can be found in Fig. 12.

2) Training Setting: We adopt an end-to-end visuomotor control policy that takes images and robot proprioceptive signals as inputs and continuously outputs robot control actions. We employ a model named Seer [55], which features an autoregressive transformer architecture. Multi-view images are processed through a MAE-pretrained ViT encoder, and the features of robot proprioceptive states are extracted using an MLP. These features are subsequently concatenated into tokens. The information of these tokens are then integrated by a transformer encoder. The transformer encoder utilizes an autoregressive method to generate latent codes for controlling upper arm joints, dexterous hand movements, and height commands. The final control action output is generated by three distinct regression heads. The whole network are optimized using SmoothL1 loss. In real-world training scenarios, we configure the sequence length to 7, with both visual foresight and action prediction steps set to 3. We employ the MAE pretrained ViT-B encoder, using bfloat16 configuration to speed up inference. This model is trained on eight A100 GPUs for 40 epoches, and we selected the checkpoint with the lowest average validation loss for evaluation.

TABLE V: Success Rate of Imitation Learning Tasks

Tasks	Squat Pick	Pick & Place
Success Rate (%)	73.3	80.0

3) Learning Results: After training with collected data, we deploy the trained model to humanoid robot in the real world, with the trained π_{auto} taking over operator to control the robot. We employ an Nvidia RTX 4080 to run the trained

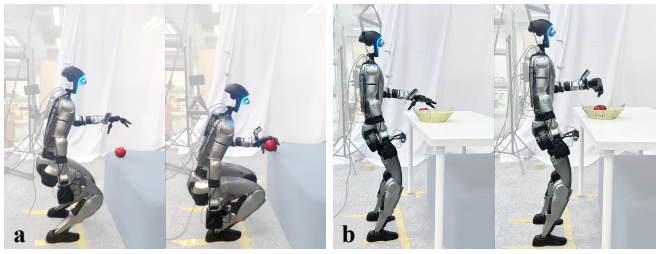


Fig. 13: **Autonomous policy controlling robot to perform tasks.** a: Squat Pick; b: Pick & Place.

model and send the output to robot. The detailed deployment configuration are introduced in Appendix C. For evaluation, we adopt the metric Success Rate (SR) of each task. After testing each proposed task for 15 times, we report the result as task success rate in Tab. V. This result shows that data collected by our teleopertion system can actually drive robots to complete complex whole-body loco-manipulation tasks. Robots that controlled by π_{auto} to perform proposed tasks are shown in Fig. 13.

V. Conclusion and Limitations

In this paper, we introduce HOMIE, a novel humanoid teleoperation cockpit for humanoid loco-manipulation. With a low-cost isomorphic exoskeleton hardware system and a humanoid loco-manipulation policy trained by our RL training framework, HOMIE enables a single operator to teleoperate the whole body of humanoid robots and perform diverse loco-manipulation tasks either in the real world or in the simulation. Owing to the incorporation of an upper-body pose curriculum, a height-tracking reward, and symmetry-based techniques, Our training framework enables the development of robust loco-manipulation policies, ensuring stable walking and squatting capabilities across diverse robotic platforms, even under dynamically changing upper-body poses. Leveraging isomorphic exoskeleton arms, HOMIE enables significantly faster task execution than other systems, and our gloves are compatible with multiple kinds of dexterous hands. We present several ablation studies and real-world experiments to validate the robustness and accuracy of our system. In addition, we show the usability of collected data for IL.

Limitations Although we have developed robust loco-manipulation policies, these policies still fall short of ensuring reliable traversal over diverse terrains. Additionally, the 15-DoF design of the motion-sensing gloves for the thumb does not fully align with human anatomy, resulting in less intuitive and smooth operation when controlling certain dexterous robotic hands. Furthermore, the current system lacks force feedback, which limits its effectiveness in applications requiring precise haptic interaction. Addressing these limitations will be a central focus of our future research efforts.

References

[1] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback. *arXiv preprint arXiv:2407.01512*, 2024.

[2] Shiqi Yang, Minghuan Liu, Yuzhe Qin, Runyu Ding, Jialong Li, Xuxin Cheng, Ruihan Yang, Sha Yi, and Xiaolong Wang. Ace: A cross-platform visual-exoskeletons system for low-cost dexterous teleoperation. *arXiv preprint arXiv:2408.11805*, 2024.

[3] Yanjie Ze, Zixuan Chen, Wenhao Wang, Tianyi Chen, Xialin He, Ying Yuan, Xue Bin Peng, and Jiajun Wu. Generalizable humanoid manipulation with improved 3d diffusion policies. *arXiv preprint arXiv:2410.10803*, 2024.

[4] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. *arXiv preprint arXiv:2307.04577*, 2023.

[5] Aadhithya Iyer, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024.

[6] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. Humanplus: Humanoid shadowing and

imitation from humans. In *Conference on Robot Learning (CoRL)*, 2024.

[7] Tairan He, Zhengyi Luo, Wenli Xiao, Chong Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Learning human-to-humanoid real-time whole-body teleoperation. *arXiv preprint arXiv:2403.04436*, 2024.

[8] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. *arXiv preprint arXiv:2406.08858*, 2024.

[9] Xuxin Cheng, Yandong Ji, Junming Chen, Ruihan Yang, Ge Yang, and Xiaolong Wang. Expressive whole-body control for humanoid robots. *arXiv preprint arXiv:2402.16796*, 2024.

[10] Mazeyu Ji, Xuanbin Peng, Fangchen Liu, Jialong Li, Ge Yang, Xuxin Cheng, and Xiaolong Wang. Exbody2: Advanced expressive humanoid whole-body control. *arXiv preprint arXiv:2412.13196*, 2024.

[11] Chenhao Lu, Xuxin Cheng, Jialong Li, Shiqi Yang, Mazeyu Ji, Chengjing Yuan, Ge Yang, Sha Yi, and Xiaolong Wang. Mobile-television: Predictive motion priors for humanoid whole-body control. *arXiv preprint arXiv:2412.07773*, 2024.

[12] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, October 2019.

[13] Chen Wang, Haochen Shi, Weizhuo Wang, Ruohan Zhang, Li Fei-Fei, and C Karen Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

[14] Zipeng Fu, Tony Z. Zhao, and Chelsea Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[15] Philipp Wu, Yide Shentu, Zhongke Yi, Xingyu Lin, and Pieter Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators, 2023.

[16] Hongjie Fang, Hao-Shu Fang, Yiming Wang, Jieji Ren, Jingjing Chen, Ruo Zhang, Weiming Wang, and Cewu Lu. Airexo: Low-cost exoskeletons for learning whole-arm manipulation in the wild. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15031–15038. IEEE, 2024.

[17] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[18] Toru Lin, Yu Zhang, Qiyang Li, Haozhi Qi, Brent Yi, Sergey Levine, and Jitendra Malik. Learning visuotactile skills with two multifingered hands. *arXiv preprint arXiv:2404.16823*, 2024.

[19] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[20] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic Telekinesis: Learning a Robotic Hand Imitator by Watching Humans on YouTube. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022.

[21] Jinhan Li, Yifeng Zhu, Yuqi Xie, Zhenyu Jiang, Mingyo Seo, Georgios Pavlakos, and Yuke Zhu. Okami: Teaching humanoid robots manipulation skills through single video imitation. In *8th Annual Conference on Robot Learning*, 2024.

[22] Manuel Caeiro-Rodríguez, Iván Otero-González, Fernando A Mikic-Fonte, and Martín Llamas-Nistal. A systematic review of commercial smart gloves: Current status and applications. *Sensors*, 21(8):2667, 2021.

[23] Hangxin Liu, Zhenliang Zhang, Xu Xie, Yixin Zhu, Yue Liu, Yongtian Wang, and Song-Chun Zhu. High-fidelity grasping in virtual reality using a glove-based system. In *2019 international conference on robotics and automation (icra)*, pages 5180–5186. IEEE, 2019.

[24] Hangxin Liu, Xu Xie, Matt Millar, Mark Edmonds, Feng Gao, Yixin Zhu, Veronica J Santos, Brandon Rothrock, and Song-Chun Zhu. A glove-based system for studying hand-object manipulation via joint pose and force sensing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6617–6624. IEEE, 2017.

[25] Nepyope. Project-homunculus. https://github.com/nepyope/Project-Homunculus, 2023. Last accessed: Sep. 18, 2024.

[26] Stefano Dafarra, Ugo Pattacini, Giulio Romualdi, Lorenzo Rapetti, Riccardo Grieco, Kourosh Darvish, Gianluca Milani, Enrico Valli, Ines Sorrentino, Paolo Maria Viceconte, et al. icub3 avatar system: Enabling remote fully immersive embodiment of humanoid robots. *Science Robotics*, 9(86):eadh3834, 2024.

[27] Hirofumi Miura and Isao Shimoyama. Dynamic walk of a biped. *The International Journal of Robotics Research*, 3(2):60–74, 1984.

[28] Patrick M Wensing, Michael Posa, Yue Hu, Adrien Escande, Nicolas Mansard, and Andrea Del Prete. Optimization-based control for dynamic legged robots. *IEEE Transactions on Robotics*, 2023.

[29] Federico L Moro and Luis Sentis. Whole-body control of humanoid robots. *Humanoid robotics: a reference*, pages 1161–1183, 2019.

[30] Qiang Zhang, Peter Cui, David Yan, Jingkai Sun, Yiqun Duan, Gang Han, Wen Zhao, Weining Zhang, Yijie Guo, Arthur Zhang, et al. Whole-body humanoid robot locomotion with human reference. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11225–11231. IEEE, 2024.

[31] Matthew Chignoli, Donghyun Kim, Elijah Stanger-Jones, and Sangbae Kim. The mit humanoid robot: Design,

motion planning, and control for acrobatic behaviors. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 1–8. IEEE, 2021.

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

[33] Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Ri-Zhao Qiu, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.

[34] Guoping Pan, Qingwei Ben, Zhecheng Yuan, Guangqi Jiang, Yandong Ji, Jiangmiao Pang, Houde Liu, and Huazhe Xu. Roboduet: A framework affording mobile-manipulation and cross-embodiment. *arXiv preprint arXiv:2403.17367*, 2024.

[35] Tifanny Portela, Andrei Cramariuc, Mayank Mittal, and Marco Hutter. Whole-body end-effector pose tracking. *arXiv preprint arXiv:2409.16048*, 2024.

[36] Huy Ha, Yihuai Gao, Zipeng Fu, Jie Tan, and Shuran Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. *arXiv preprint arXiv:2407.10353*, 2024.

[37] Xinyang Gu, Yen-Jen Wang, Xiang Zhu, Chengming Shi, Yanjiang Guo, Yichen Liu, and Jianyu Chen. Advancing humanoid locomotion: Mastering challenging terrains with denoising world model learning. *arXiv preprint arXiv:2408.14472*, 2024.

[38] Junfeng Long, Junli Ren, Moji Shi, Zirui Wang, Tao Huang, Ping Luo, and Jiangmiao Pang. Learning humanoid locomotion with perceptive internal model, 2024.

[39] Zixuan Chen, Xialin He, Yen-Jen Wang, Qiayuan Liao, Yanjie Ze, Zhongyu Li, S Shankar Sastry, Jiajun Wu, Koushil Sreenath, Saurabh Gupta, et al. Learning smooth humanoid locomotion through lipschitz-constrained policies. *arXiv preprint arXiv:2410.11825*, 2024.

[40] Pranay Dugar, Aayam Shrestha, Fangzhou Yu, Bart van Marum, and Alan Fern. Learning multi-modal whole-body control for real-world humanoid robots. *arXiv preprint arXiv:2408.07295*, 2024.

[41] Zhongyu Li, Xue Bin Peng, Pieter Abbeel, Sergey Levine, Glen Berseth, and Koushil Sreenath. Reinforcement learning for versatile, dynamic, and robust bipedal locomotion control. *The International Journal of Robotics Research*, page 02783649241285161, 2024.

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

[43] Qiayuan Liao, Bike Zhang, Xuanyu Huang, Xiaoyu Huang, Zhongyu Li, and Koushil Sreenath. Berkeley humanoid: A research platform for learning-based control. *arXiv preprint arXiv:2407.21781*, 2024.

[44] Chong Zhang, Wenli Xiao, Tairan He, and Guanya Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024.

[45] Ziwen Zhuang, Shenzhe Yao, and Hang Zhao. Humanoid parkour learning. In *8th Annual Conference on Robot Learning*, 2024.

[46] Zhaoyuan Gu, Junheng Li, Wenlan Shen, Wenhao Yu, Zhaoming Xie, Stephen McCrory, Xianyi Cheng, Abdulaziz Shamsah, Robert Griffin, C Karen Liu, et al. Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning. *arXiv preprint arXiv:2501.02116*, 2025.

[47] Zhengyi Luo, Jinkun Cao, Josh Merel, Alexander Winkler, Jing Huang, Kris Kitani, and Weipeng Xu. Universal humanoid motion representations for physics-based control. *arXiv preprint arXiv:2310.04582*, 2023.

[48] Junfeng Long, Zirui Wang, Quanyi Li, Liu Cao, Jiawei Gao, and Jiangmiao Pang. Hybrid internal model: Learning agile legged locomotion with simulated robot response. In *The Twelfth International Conference on Learning Representations*, 2024.

[49] Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

[50] Zhi Su, Xiaoyu Huang, Daniel Ordoñez-Apraez, Yunfei Li, Zhongyu Li, Qiayuan Liao, Giulio Turrisi, Massimiliano Pontil, Claudio Semini, Yi Wu, et al. Leveraging symmetry in rl-based legged locomotion control. *arXiv preprint arXiv:2403.17320*, 2024.

[51] Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[52] Gabriel B Margolis and Pulkit Agrawal. Walk these ways: Tuning robot control for generalization with multiplicity of behavior. In *Conference on Robot Learning*, pages 22–31. PMLR, 2023.

[53] Hanqing Wang, Jiahe Chen, Wensi Huang, Qingwei Ben, Tai Wang, Boyu Mi, Tao Huang, Siheng Zhao, Yilun Chen, Sizhe Yang, et al. Grutopia: Dream general robots in a city at scale. *arXiv preprint arXiv:2407.10943*, 2024.

[54] Mayank Mittal, Calvin Yu, Qinxi Yu, Jingzhou Liu, Nikita Rudin, David Hoeller, Jia Lin Yuan, Ritvik Singh, Yunrong Guo, Hammad Mazhar, Ajay Mandlekar, Buck Babich, Gavriel State, Marco Hutter, and Animesh Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[55] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *arXiv preprint arXiv:2412.15109*, 2024.

APPENDIX A
RL TRAINING AND EVALUATION DETAILS.

A. Network Architecture

We use a network architecture similar to that employed in a previous quadruped locomotion work called HIM [48]. The architecture consists of three main components: an estimator network $\mathcal{E}$, a follow-up network $\mathcal{N}$ that takes the output of $\mathcal{E}$ as part of its input, and a critic network $\mathcal{C}$. Together, $\mathcal{E}$ and $\mathcal{N}$ form the actor module, as illustrated in Fig. 3. All networks are implemented as 3-layer multilayer perceptrons (MLPs). Below, we describe their specific architectures, where $N_{joints} = N_{lower} + N_{upper}$, with N_{lower} representing the number of lower-body joints and N_{upper} the number of upper-body joints of the robot. Given that the dimensions of the input vectors are as follows: $Dim([C_t, \omega_t, g_t]) = 4 + 3 + 3 = 10$ and $Dim([v_x, v_y, \omega_{yaw}]) = 3$, we can determine the input dimensions for each network. The workflow is as follows: The sequence $O_{t-5:t}$ is fed into the encoder of $\mathcal{E}$. The encoder processes O_t along with the ground truth values $([v_x, v_y, \omega_{yaw}])$ and outputs the encoded information $\hat{I}_t$. $\hat{I}_t$ is then passed to $\mathcal{N}$ and is also used for contrastive learning with the output $\hat{I}'_t$ of the target network. Combining $\hat{I}_t$ and O_t, $\mathcal{N}$ produces the action a_t, whose dimension equals N_{lower}. The critic network $\mathcal{C}$ takes O_t and the ground truth $([v_x, v_y, \omega_{yaw}])$ as its input.

Listing 1: Architecture of Neural Networks Used by Our RL Training Framework.

```
(ℰ): HIMEstimator(
   (encoder): Sequential(
     (0): Linear(in_features=6 × (10 + 2 × N_joints + N_lower), out_features=256, bias=True)
     (1): ELU(alpha=1.0)
     (2): Linear(in_features=256, out_features=256, bias=True)
     (3): ELU(alpha=1.0)
     (4): Linear(in_features=256, out_features=35, bias=True)
   )
   (target): Sequential(
     (0): Linear(in_features=10 + 2 × N_joints + N_lower, out_features=256, bias=True)
     (1): ELU(alpha=1.0)
     (2): Linear(in_features=256, out_features=256, bias=True)
     (3): ELU(alpha=1.0)
     (4): Linear(in_features=256, out_features=32, bias=True)
   )
   (proto): Embedding(64, 32)
 )
 (𝒩): Sequential(
   (0): Linear(in_features=35 + 10 + 2 × N_joints + N_lower, out_features=512, bias=True)
   (1): ELU(alpha=1.0)
   (2): Linear(in_features=512, out_features=256, bias=True)
   (3): ELU(alpha=1.0)
   (4): Linear(in_features=256, out_features=256, bias=True)
   (5): ELU(alpha=1.0)
   (6): Linear(in_features=256, out_features=N_lower, bias=True)
 )
 (𝒞): Sequential(
   (0): Linear(in_features=3 + 10 + 2 × N_joints + N_lower, out_features=512, bias=True)
   (1): ELU(alpha=1.0)
   (2): Linear(in_features=512, out_features=256, bias=True)
   (3): ELU(alpha=1.0)
   (4): Linear(in_features=256, out_features=256, bias=True)
   (5): ELU(alpha=1.0)
   (6): Linear(in_features=256, out_features=1, bias=True)
 )
```

B. Reward Scales

We present the reward functions and their corresponding scales used in training the Unitree G1 and Fourier GR-1 robots in Tab. VI. Our reward functions adapt from PIM [38], a previous work on humanoid locomotion, with several modifications and new additions. In addition to the "Squat knee" term r_{knee} introduced in Sec. III-B3, we revise the "Base height tracking" reward to align it with other tracking terms, as our objective is to track the changing target base height. Furthermore, we decompose the linear velocity tracking reward into separate components for x and y velocity tracking to better distinguish the robot's performance in these directions. To encourage the robot to remain stationary when zero velocity is required, we introduce a "stand still" reward. However, we observe that this reward can cause the robot to become overly inclined to maintain stillness,

leading to instability during transitions between stationary and moving states. Experimental results indicate that reducing the K_p of the ankle joint alleviates this issue. The rationale is that a lower K_p tends to produce smaller torque outputs, making the ankle joint more responsive to positional changes when the center of gravity shifts. This positional change provides proprioceptive feedback to the policy, prompting necessary adjustments. The reward scales used in the training processes of these two heterogeneous robots are largely similar, further demonstrating the generality of our framework.

TABLE VI: Reward Functions and Weights Used to Train Low-manipulation Policy

Reward	Equation	Weight of Unitree G1	Weight of Fourier GR-1
x Vel. tracking	$\exp\left\{-4\|\mathbf{v}_x - \mathbf{v}_{r,x}\|_2^2\right\}$	1.5	1.5
y Vel. tracking	$\exp\left\{-4\|\mathbf{v}_y - \mathbf{v}_{r,y}\|_2^2\right\}$	1.0	1.0
Ang. Vel. tracking	$\exp\left\{-4\left\|\omega_{\text{yaw}} - \omega_{\text{r,yaw}}\right\|^2\right\}$	2.0	1.0
Base height tracking	$\exp\left\{-4\left\|h_{\text{t}} - h_{\text{r,t}}\right\|^2\right\}$	2.0	2.0
Lin. Vel. z	$v_{r,z}^2$	-0.5	-0.5
Ang. Vel. xy	$\|\boldsymbol{\omega}_{r,xy}\|_2^2$	-0.025	-0.025
Orientation	$\|\mathbf{g}_x\|_2^2 + \|\mathbf{g}_y\|_2^2$	-1.5	-1.5
Action rate	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|_2^2$	-0.01	-0.01
Hip joint deviation	$\sum_{\text{hip joints}} \|\theta_i - \theta_i^{default}\|^2$	-0.2	-0.5
Ankle joint deviation	$\sum_{\text{ankle joints}} \|\theta_i - \theta_i^{default}\|^2$	-0.5	-0.75
Squat knee	$-\|(h_{r,t} - h_t) \times (\frac{q_{knee,t} - q_{knee,min}}{q_{knee,max} - q_{knee,min}} - \frac{1}{2})\|$	-0.75	-0.75
Dof Acc.	$\sum_{\text{all joints}} \frac{\|\dot{q}_{t,i} - \dot{q}_{t-1,i}\|^2}{dt}$	-2.5×10^{-7}	-2.5×10^{-7}
Dof pos limits	$\sum_{\text{all joints}} \text{out}_i$	-2.0	-2.0
Feet air time	$\mathbf{1}_{\{\text{first contact}\}}(T_{air} - 0.5)$	0.05	0.05
Feet clearance	$\sum_{feet} \left(p_z^{\text{target}} - p_z^i\right)^2 \cdot v_{xy}^i$	-0.25	-0.25
Feet lateral distance	$\|y^B_{\text{left foot}} - y^B_{\text{right foot}}\| - d_{min}$	0.5	0.5
Knee lateral distance	$\|y^B_{\text{left knee}} - y^B_{\text{right knee}}\| - d_{min}$	1.0	1.0
Feet ground parallel	$\sum_{feet} Var(H_i)$	-2.0	-2.0
Feet parallel	$Var(D)$	-3.0	-3.0
Smoothness	$\|\mathbf{a}_t - 2\mathbf{a}_{t-1} + \mathbf{a}_{t-2}\|_2^2$	-0.05	-0.05
Joint power	$\frac{\|\boldsymbol{\tau}\|\dot{\boldsymbol{\theta}}\|^T}{\|\mathbf{v}\|_2^2 + 0.2 * \|\boldsymbol{\omega}\|_2^2}$	-2.0×10^{-5}	-2.0×10^{-5}
Feet stumble	$\mathbf{1}\left\{\exists i, \left\|\mathbf{F}_i^{xy}\right\| > 3\left\|F_i^z\right\|\right\}$	-1.5	-1.5
Torques	$\sum_{\text{all joints}} \|\frac{\tau_i}{kp_i}\|_2^2$	-2.5×10^{-6}	-2.5×10^{-6}
Dof Vel.	$\sum_{\text{all joints}} \dot{\theta}_i\|_2^2$	-1×10^{-4}	-1×10^{-4}
Dof Vel. limit	$\sum_{\text{all joints}} RELU(\hat{\theta}_i - \hat{\theta}_i^{max})$	-2×10^{-3}	-2×10^{-3}
Torque limits	$\sum_{\text{all joints}} RELU(\hat{\tau}_i - \hat{\tau}_i^{max})$	-0.1	-0.2
No fly	$\mathbf{1}\{\text{only one feet on ground}\}$	0.75	0.5
Joint tracking error	$\sum_{\text{all joints}} \|\theta_i - \theta_i^{target}\|^2$	-0.1	-0.25
Feet slip	$\sum_{feet} \left\|\mathbf{v}_i^{\text{toot}}\right\| * \sim \mathbf{1}_{\text{new contact}}$	-0.25	-0.25
Feet contact force	$\sum_{feet} RELU(F_i^z - F_{th})$	-2.5×10^{-4}	-2.5×10^{-4}
Contact momentum	$\sum_{feet} \|v_i^z * F_i^z\|$	2.5×10^{-4}	2.5×10^{-4}
Action vanish	$\sum_{\text{all joints}} (\max\{0, a_{i,t} - a_{i,max}\} + \min\{0, a_{i,min} - a_{i,t}\})$	-1.0	-1.0
Stand still	$Num_{\{feet\ not\ on\ ground\}} \times \mathbf{1}_{stand\ still}$	-0.15	-0.2

C. Domain Randomization

To improve the robustness of the trained policy, we employ domain randomization to simulate several kinds of random noises that may occur while deploying in the real world. The terms used for randomization, along with their descriptions and ranges, are listed in Tab. VII. Specifically, we introduce a term to randomize the mass of the hands, enhancing the robot's capability to hold objects effectively. The ranges for the Unitree G1 and Fourier GR-1 are the same.

TABLE VII: Randomization Terms, Description, and Ranges

Term	Description	Ranges
Actuation offset $(N \cdot m)$	Random torque offsets applied to the computed motor torques	$[-0.05, 0.05]$
Torso payload mass (Kg)	Additional random mass attached to the torso and hand links	$[-5.00, 10.00]$
Hand payload mass (Kg)	Additional random mass attached to the hand links	$[-0.10, 0.30]$
CoM displacement (m)	Random offsets applied to the center of mass (CoM) position of the torso link	$[-0.1, 0.1]$
Link mass $(-)$	Random scaling factors applied to the masses of the robot's links	$[0.80, 1.20]$
Friction coefficient $(-)$	Random friction coefficients applied to the robot's links	$[0.10, 2.00]$
Restitution $(-)$	Random restitution coefficients applied to the robot's links	$[0.00, 1.00]$
K_p (N/rad)	Random scaling factors applied to the proportional gain (K_p) of the robot's joints	$[0.90, 1.10]$
K_d $(N/(m/s))$	Random scaling factors applied to the derivative gain (K_d) of the robot's joints	$[0.90, 1.10]$
Initial joint pos scale $(-)$	Random scaling factors applied to the initial positions of the robot's joints	$[0.80, 1.20]$
Initial joint pos offset (rad)	Random offsets added to the initial positions of the robot's joints	$[-0.10, 0.10]$
Push robot (m/s)	Random x and y velocities applied to the robot to simulate external pushes	$[-0.50, 0.50]$
Dof pos obs (rad)	Random dof velocity added to the observation of joint positions	$[-0.02, 0.02]$
Dof vel obs (rad/s)	Random dof velocity added to the observation of joint velocities	$[-2.00, 2.00]$
Ang vel obs (rad/s)	Random dof velocity added to the observation of body angular velocities	$[-0.50, 0.50]$
Gravity obs (m/s^2)	Random dof velocity added to the observation of gravities projected to robot's body frame	$[-0.05, 0.05]$

D. Other Key Parameters

We list other key parameters used to train the Unitree G1 and Fourier GR-1 in Tab. VIII. The same settings are applied for both training and evaluation. Additionally, we adjust the base height target value when the environment is used to train squatting; otherwise, the robot is required to track a constant height value while walking. Terms marked with $*$ indicate that exceeding the specified range will result in penalties through corresponding rewards.

TABLE VIII: Key Parameters Used to Train Robots

Term	Unitree G1	Fourier GR-1
Height target while walking (m)	0.74	0.90
X Lin. Vel. range (m/x)	$[-0.80, 1.20]$	$[-0.80, 1.20]$
Y Lin. Vel. range (m/s)	$[-0.50, 0.50]$	$[-0.80, 0.80]$
Yaw Ang. Vel. range (rad/s)	$[-0.80, 0.80]$	$[-1.00, 1.00]$
Squat height range (m)	$[-0.24, 0.74]$	$[-0.30, 0.90]$
Soft dof pos limit scale * $(-)$	0.975	0.975
Soft dof vel limit scale * $(-)$	0.80	0.80
Soft dof torque limit scale * $(-)$	0.95	0.95
Max contact force * (N)	400.00	500.00
Least feet distance * (m)	0.20	0.20
Least knee distance * (m)	0.20	0.20
Most feet distance * (m)	0.35	0.40
Most knee distance * (m)	0.35	0.40
Clearance height target * (m)	0.14	0.15
Push interval (s)	4.00	4.00
Upper-body poses resampling interval (s)	1.00	1.00
Commands resampling interval (s)	4.00	4.00

E. Function Visualization

For better understanding of the equations we proposed in Eq. (3) and Eq. (4), we visualize them in Fig. 14. As shown in the left figure, $p(x|r_a)$ can take any value in the range $[0, 1]$ when $r_a \in [0, 1)$. When r_a is small, it is more likely to take smaller values of x. As r_a increases, the probability of taking larger values of x also increases. When $r_a \to 1$, the entire distribution becomes $\mathcal{U}(0, 1)$. In the right figure, we can observe that regardless of the position of (h, q), r_{knee} encourages $q_{knee,t}$ to change in a direction that brings $h_{r,t}$ closer to h_t. This achieves the goal of guiding the robot to track the base height by either bending or straightening its knees.

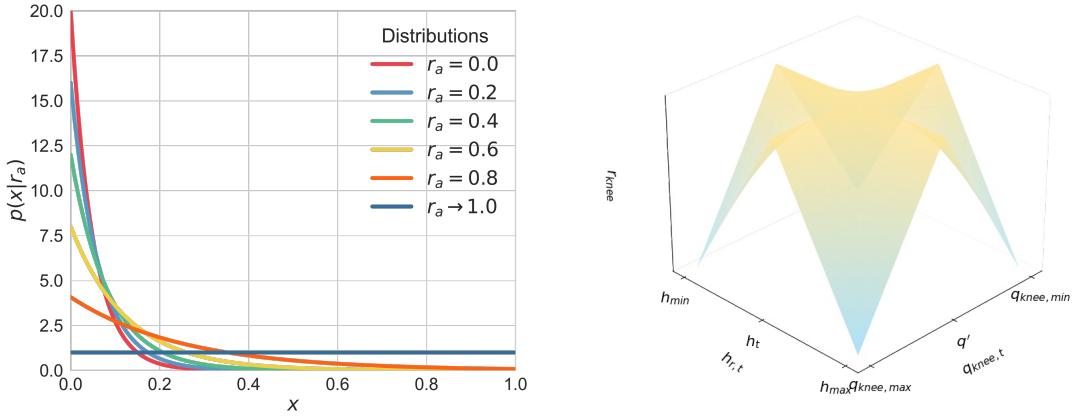


Fig. 14: Visualization of proposed functions. Left: Visualization of $p(x|r_a)$ in Eq. (3). Right: Visualization of r_{knee} in Eq. (4).

F. Terrain Traverse

In order to expand the feasible workspace of robots, we integrate our training framework with a previous humanoid locomotion method called PIM [38] to enable our robots to traverse stairs. As shown in the left figure of Fig. 15, we successfully train the Unitree G1 in Isaac Gym to traverse high stairs. However, when deploying the trained policy in the real world, as shown in the right figure of Fig. 15, the robot struggles to walk stably and collides with the stair, despite eventually stepping onto it. This instability arises because the head of the Unitree G1 cannot remain fixed, causing movement of the LiDAR mounted on it. Additionally, the elevation map acquisition method used by PIM lacks high resolution, further exacerbating the sim2real gap. In the future, we will explore methods to truly enable our robots to traverse any terrain effectively.

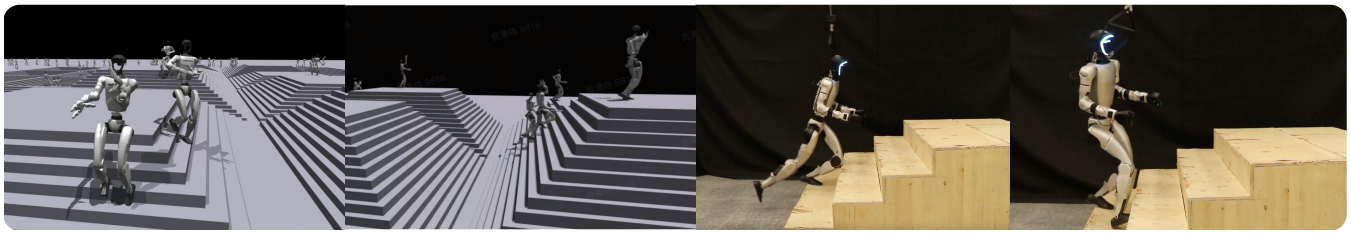


Fig. 15: Training robots to traverse stairs. Left: Training in simulation; Right: Deployment in the real world.

APPENDIX B
HARDWARE SYSTEM DETAILS

In Fig. 2, we present the hardware system design framework of HOMIE, which comprises three integral components: isomorphic exoskeleton arms, a pair of motion-sensing gloves, and a pedal. The primary structural elements of these components are fabricated using 3D printing technology with PLA basic material.

A. Isomorphic Exoskeleton Details

The Isomorphic Exoskeleton adopts a hollowed-out and mortise-and-tenon design, which not only ensure structural integrity but also significantly reduce the overall weight and assembly complexity, as well as facilitate the routing of servo motor connections. The structural components are fixed to the servos through four types of connection methods: two methods that directly connect to the servo body Fig. 16(a), (b) and two methods that connect to the servo disks Fig. 16(c) To further enhance stability and strength, additional servo disks are installed on the opposite side of the servos at certain joints, allowing the structural components to connect directly to the servo disks on both sides, as illustrated in Fig. 16(c). We present the physical models of the Isomorphic Exoskeletons adapted for the Unitree G1 and Fourier GR1 in Fig. 17(a). Due to the different configurations of the Humanoid, there are significant structural differences in the wrist and shoulder components between the two sets of Isomorphic Exoskeletons, while the other components and their usage remain identical. The two sets of Exoskeletons share the same back connector, which integrates functionalities for operator attachment, docking station fixation, U2D2 placement, and bilateral arm linkage, as illustrated in Fig. 17(b).

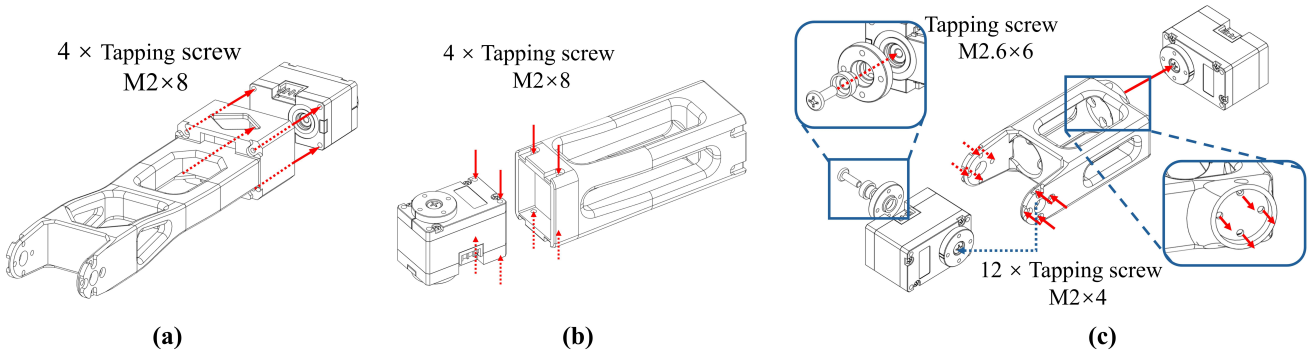


Fig. 16: The assembly methods of the servos and structural components, along with the screw requirements, are as follows: (a), (b:) The structural components are directly assembled and fixed to the servo body; (c:) The structural components are assembled and fixed to the servos via one or two servo disks, respectively.

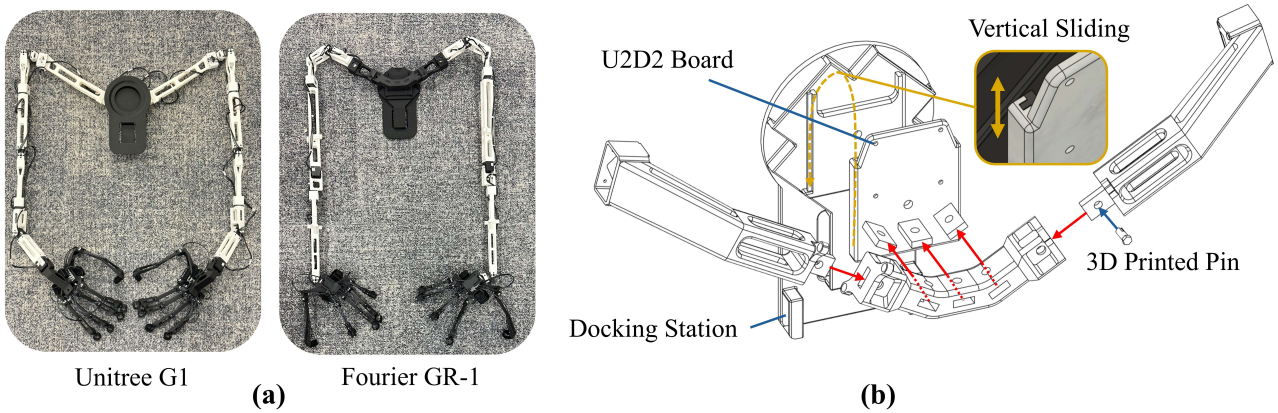


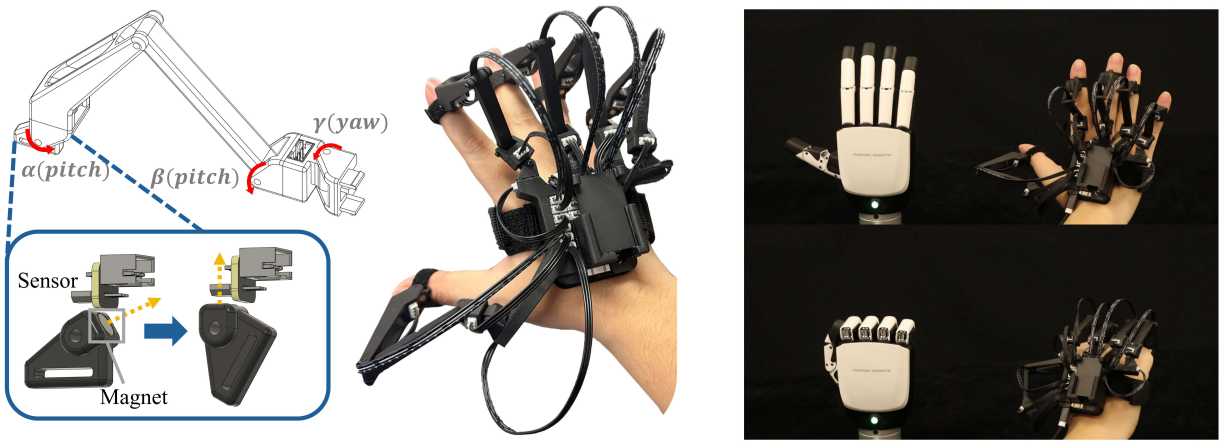
Fig. 17: (a:) Physical models of two different Humanoids' Isomorphic Exoskeletons, equipped with servos, back connectors, and motion-sensing gloves; (b:) Schematic diagram of the back connector assembly and functionality, where the connectors can be fixed using a dovetail structure and plugs. The U2D2 board and docking station are external physical components.

B. Motion-sensing Glovesc Details

The motion-sensing gloves are secured to the palm via a length-adjustable elastic strap and connected to the fingertips through five smaller length-adjustable elastic straps, which facilitate finger fixation and critical angle mapping, ensuring adaptability to operators with varying hand sizes. A microcontroller is embedded within the palm section of the gloves, featuring exposed ports that allow direct connection to the 15 Hall sensor modules located at the fingers, as depicted in Fig. 18(a). For the joint mapping angle range and acquisition accuracy of each finger, we have listed the data in TABLEx. Each finger of the glove has three degrees of freedom, which are shown in Fig. 18(a), namely the pitch motion of the fingertip (α), the pitch motion of the finger pad (β), and the yaw motion of the finger pad (γ). Due to differences in the structural length of the thumb, the pinky, and the other three fingers, we have divided them into three parts. It should be noted that the angular movement of the finger joints does not exhibit a significant linear relationship with the changes in the Hall sensor readings caused by the induced magnetic field variation. This is related to the positioning of the magnets and Hall sensors, as well as the structural design of the gloves. In our motion-sensing gloves design, the relationship between the two follows an exponential pattern within their transformation range, especially in the pitch motion of the fingers and finger pads, from open to fist. Furthermore, our gloves have undergone control testing on the Inspire Dexterous Hands RH56DFTP actual device, illustrated in Fig. 18(b).

C. Foot Pedal Details

The foot pedal consists of three small pedals and two mode-switching buttons, all fixed onto a large base plate, as shown in Fig. 5. The operator can press the small pedals, which cause the structural components to rotate, thereby driving the potentiometer at the bottom to rotate. The spring within the structure ensures that when the operator releases the pedal, it springs back, returning to the initial position, as shown in Fig. 19(a). The potentiometer we use, model 0932, has a range of angular movement of 270°, with the actual pedal movement range being 40°. As for the mode-switching buttons, the operator



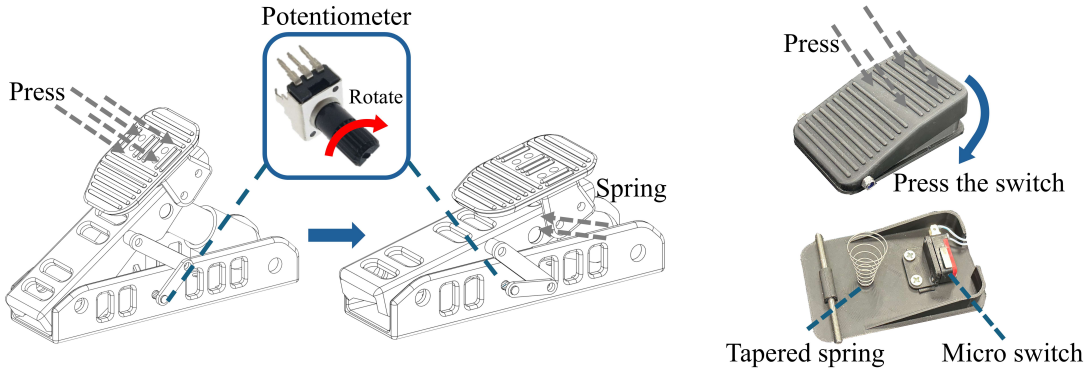
(a) (b)

Fig. 18: (a): The schematic diagrams of the three degrees of freedom for each finger and the rotation of the magnet affecting the magnetic field direction, as well as the photograph of the actual wearable device. (b): Physical images of the Inspire Hands actual device in both open and clenched fist states.

TABLE IX: 15 DoF Joint Mapping Angle Range and Acquisition Accuracy. (Acc:accuracy)

Term	Description	Angle Range	Acquisition Range	Acquisition Acc.
α_{thumb}	Pitch motion of the thumb tip	65°	528 units	0.123°/unit
β_{thumb}	Pitch motion of the thumb pad	100°	1024 units	0.098°/unit
γ_{thumb}	Yaw motion of the thumb pad	90°	832 units	0.108°/unit
α_{pinky}	Pitch motion of the pinky tip	70°	880 units	0.080°/unit
β_{pinky}	Pitch motion of the pinky pad	90°	1136 units	0.079°/unit
γ_{pinky}	Yaw motion of the pinky pad	45°	416 units	0.108°/unit
α_{other}	Pitch motion of the index, middle, and ring finger tips	70°	928 units	0.075°/unit
β_{other}	Pitch motion of the index, middle, and ring finger pads	90°	1072 units	0.088°/unit
γ_{other}	Yaw motion of the index, middle, and ring finger pads	40°	512 units	0.078°/unit

can press the buttons on the surface, which will cause a change in the high and low levels of the micro switch, as shown in Fig. 19(b). The function of the tapered spring is the same as the spring in the small pedals.



(a) (b)

Fig. 19: (a): Schematic diagram of the small pedal principle, where the operator's foot press drives the potentiometer to rotate. (b): Schematic diagram of the mode-switching buttons, where the operator's foot press changes the state of the micro switch.

APPENDIX C
DEPLOYMENT DETAILS

A. System Deployment

We deploy our trained policy π_{loco} directly onto the Unitree G1's onboard computing unit—a Nvidia Jetson Orin capable of 275 TOPS—allowing π_{loco} to run at 50 Hz using the robot's state information to control walking and squatting, matching the frequency used during Isaac Gym training. We use an isomorphic exoskeleton-based approach to control the robot, as shown in Fig. 20(b). A CPU-only host computer connects via four data lines to the isomorphic arm, left and right gloves, and the pedal's microcontroller, reading real-time data. It then transmits q_{upper} and C_t to the G1 over Wi-Fi via TCP, enabling the robot to set upper-body poses and compute lower-body actions a_t through π_{loco} for full-body control, while simultaneously returning real-time images to the host. The D455 camera provides 640×480 images at roughly 30Hz over TCP. Due to hardware constraints and TCP network limitations, the G1 cannot directly process high-frequency data. In our deployment, we therefore update the arm joint position targets at 10Hz and interpolate these targets 50 times to drive the robot's upper body smoothly. However, if a robot can accept higher-frequency control signals, our system can support an update frequency over 200Hz.

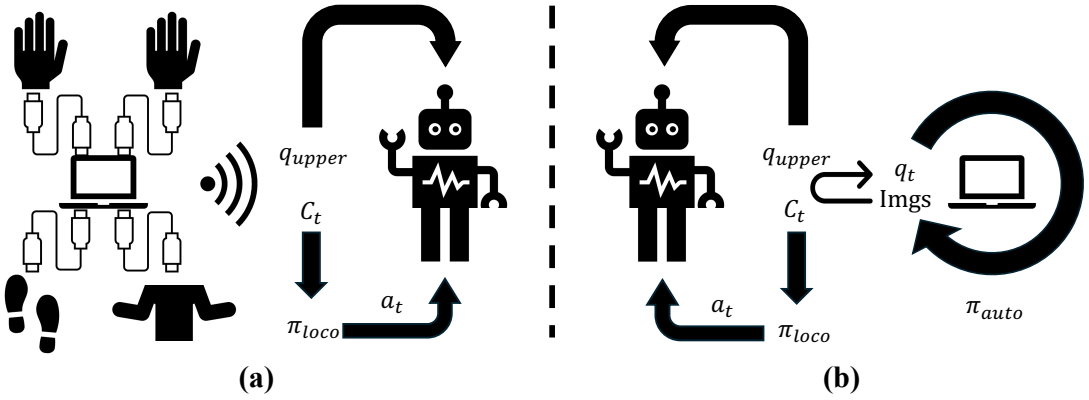


Fig. 20: Deployment of our system with cockpit and autonomous policy. (a): Deployment with robot controlled by isomorphic exoskeleton cockpit. (b): Deployment with robot controlled by autonomous policy π_{auto}.

B. IL Deployment

Once we train the policy π_{auto} using the method described in Sec. IV-D2, we deploy it as illustrated in Fig. 20(b). We connect the Unitree G1 to a host equipped with an Nvidia RTX 4080 GPU via an Ethernet cable, enabling wired TCP communication. This wired setup provides the faster data transfer needed for transmitting images. The runtime involves two processes on the G1, labeled 1 and 2, and one process on the host, labeled 3. Process 1 captures images from the D455 camera mounted on the G1's head and from the D435 cameras on its arms, along with the robot's joint information $q_{t,i}$. This data is then sent to the host. Process 3 receives the inputs, performs inference of π_{auto} to compute C_t and q_{upper}, and returns these results to the G1. Finally, process 2 controls the robot's motion using the inferred commands. The entire loop runs at a frequency of 10Hz.

C. Simulation Deployment

We train the policies in Isaac Gym, which is sufficient for training locomotion policies but lacks the capability to simulate realistic scenes. Therefore, we employ a sim2sim process to transfer our policies to Isaac Sim. The core of this process involves aligning the joint order and quaternion conventions between the two platforms. In Isaac Gym, the joint order follows depth-first ordering, and quaternions are formatted as xyzw. In contrast, Isaac Sim uses breadth-first ordering for joints and wxyz for quaternions [54]. As a result, both the neural network's observation $O_{t-5:t}$ and the computed action a_t must undergo corresponding order adjustments. To simulate the real-world camera perspective in the simulation, we directly add a Camera to the robot's USD file and place its prim path under the prim path of the link to which it is bound. This ensures that the camera moves along with the corresponding link during motion in the simulation. The camera parameters, such as resolution and focal length, are configured to match those used in the real world.