REFINE-DP: Diffusion Policy Fine-tuning for Humanoid Loco-manipulation via Reinforcement Learning

Zhaoyuan Gu*, Yipu Chen*, Zimeng Chai*, Alfred Cueva, Thong Nguyen†, Yifan Wu†, Huishu Xue†, Amelie Minji Kim, Isaac Legene, Yongxin Chen, Ye Zhao

***Abstract*—Humanoid loco-manipulation requires coordinated high-level motion plans with stable, low-level whole-body execution under complex robot-environment dynamics and long-horizon tasks. While diffusion policies (DPs) show promise for learning from demonstrations, deploying them on humanoids poses critical challenges: the motion planner trained offline is decoupled from the low-level controller, leading to poor command tracking, compounding distribution shift, and task failures. The common approach of scaling demonstration data is prohibitively expensive for high-dimensional humanoid systems. To address this challenge, we present REFINE-DP (REinforcement learning FINE-tuning of Diffusion Policy), a hierarchical framework that jointly optimizes a DP high-level planner and an RL-based low-level loco-manipulation controller. The DP is fine-tuned via a PPO-based diffusion policy gradient to improve task success rate, while the controller is simultaneously updated to accurately track the planner's evolving command distribution, reducing the distributional mismatch that degrades motion quality. We validate REFINE-DP on a humanoid robot performing loco-manipulation tasks, including door traversal and long-horizon object transport. REFINE-DP achieves an over 90% success rate in simulation, even in out-of-distribution cases not seen in the pre-trained data, and enables smooth autonomous task execution in real-world dynamic environments. Our proposed method substantially outperforms pre-trained DP baselines and demonstrates that RL fine-tuning is key to reliable humanoid loco-manipulation. https://refine-dp.github.io/REFINE-DP/**

***Index Terms*—Diffusion Policy, Fine-tuning, Joint Optimization, Humanoid Robot, Locomotion and Manipulation.**

I. Introduction

Humanoid robots are increasingly capable in their physical behaviors, yet they still lack the high-level intelligence required to perform human-like tasks, such as coordinating locomotion and manipulation for door traversal. Recent studies have demonstrated agile whole-body control with both model-based methods [1] and learning-based motion imitation [2], [3]. Despite these advances, existing approaches often struggle to operate autonomously in dynamic environments, with reliable execution still depending significantly on human supervision or heuristic planning. To perform physical tasks reliably, humanoid robots must interact with the physical world through contact, such as manipulating objects and exerting force on the environment, where modeling errors and contact uncertainties frequently lead to task failure. These challenges require real-time adaptation beyond motion tracking control.

To address these challenges, recent work has explored learning high-level action generation directly from demonstrations, enabling robots to acquire task-level autonomy beyond predefined control policies. For instance, recent advances in diffusion policies (DPs) [4] have introduced a promising behavior cloning (BC) approach for learning action-generative models from expert demonstrations. These models are effective at capturing complex and multimodal expert behaviors. However, their success often relies on offline datasets, which can lead to poor out-of-distribution robustness due to distribution shift and compounding errors. To mitigate this issue, DPs are typically trained on large datasets that require high-capacity models, such as transformers, to improve coverage and generalization. However, for high-dimensional humanoid systems, this scaling strategy incurs prohibitive costs in both data collection and policy training computation, while still failing to achieve robust performance in loco-manipulation, where long horizons, high dimensionality, and compounding execution errors exacerbate distribution mismatch.

Rather than scaling data, this paper addresses the distribution-shift issue through reinforcement learning fine-tuning (RLFT) [5], [6], enabling adaptation beyond offline pre-training. Specifically, we fine-tune a DP by exploring interactions in a simulator and collecting trials to directly update the DP via policy gradient [5]. By exploring unseen state–action pairs, RLFT significantly increases the task success rate with only a sparse reward. Additionally, we demonstrate that RLFT enables a pre-trained DP on a small dataset to outperform those trained on substantially larger offline datasets.

Deploying DP on a humanoid robot remains a challenge due to the complexity of whole-body control. To simplify the action space of DP, we adopt a hierarchical framework. Specifically, the DP outputs high-level commands, such as base velocity and hand poses, while an RL-based low-level controller is responsible for loco-manipulation: maintaining locomotion stability and manipulation accuracy. This hierarchical framework keeps the high-level command space compact and intuitive, while delegating the complexity of whole-body control to a dedicated low-level policy. An illustration of the proposed hierarchical framework is in Fig. 1.

Based on this hierarchical framework, we introduce REFINE-DP (**RE**inforcement learning **FINE**-tuning of **D**iffusion **P**olicy), a novel fine-tuning approach that jointly optimizes both the DP planner and the loco-manipulation RL controller. During joint optimization, the parameters of both the DP and the low-level control policy are updated. This joint optimization maintains distributional consistency between the planner's outputs and the controller's inputs, resulting in improved command-tracking performance and higher task success rates.

The authors are with The Institute for Robotics and Intelligent Machines, Georgia Institute of Technology. (* † equally contributed)

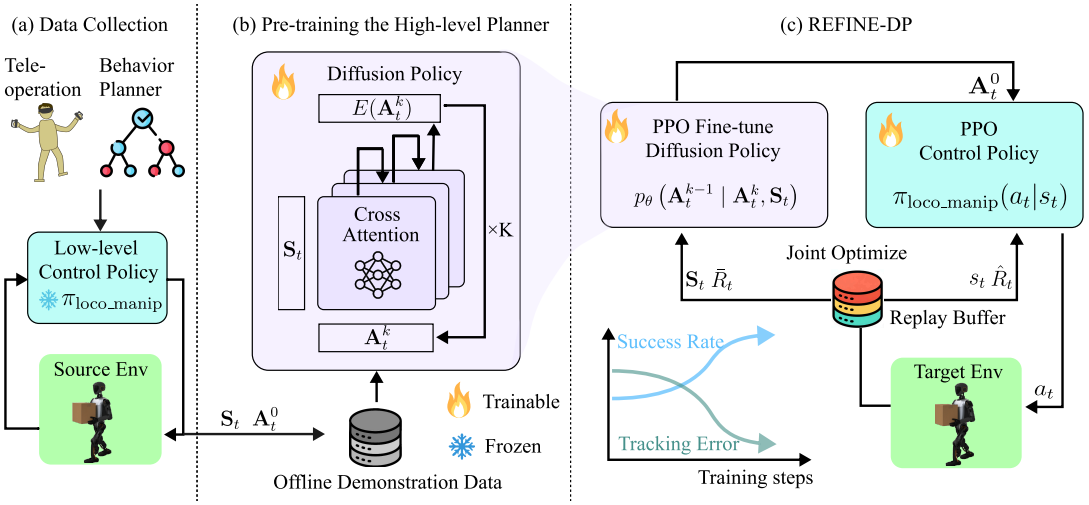

Fig. 1: Our pipeline consists of three stages: (a) Data Collection, where expert demonstrations are collected in a source environment using a frozen RL-based loco-manipulation controller; (b) Pre-training, where a diffusion policy is trained with the expert dataset containing human-demonstrated skills; and (c) Joint Optimization, where the pre-trained diffusion policy is jointly fine-tuned with the low-level controller in a target environment, enabling the robot to refine its loco-manipulation skills through RL to achieve a higher success rate and better motion tracking quality.

In summary, our main contributions are as follows:

- We design a hierarchical humanoid loco-manipulation framework in which a DP serves as the high-level autonomous planner and an RL policy acts as the low-level whole-body controller. Instead of operating directly in full-body configuration spaces, our DP outputs low-dimensional, task-relevant commands (i.e., base velocities and hand poses) tailored to humanoid loco-manipulation. These commands substantially simplify the action space of the DP, and can be easily teleoperated by human operators and executed by the RL controller.
- We introduce an RLFT approach that jointly optimizes the high-level diffusion planner and the low-level RL controller. Our method outperforms both a pre-trained DP in terms of task success rate and a fine-tuned DP alone in terms of motion tracking accuracy.
- We validate our framework on the T1 humanoid robot, enabling fully autonomous loco-manipulation across a range of tasks. These include walking and door opening, long-horizon box transport, and stepping onto an elevated platform to retrieve an object.

II. Related Works

A. Sim-to-Real RL for Humanoid Loco-manipulation Control

Loco-manipulation skills involve simultaneous locomotion and manipulation, posing a high-dimensional control problem for humanoid robots [7]. Prior work has addressed this challenge using sim-to-real reinforcement learning (RL), such as for whole-body control frameworks [8], while other approaches decompose the problem into upper- and lower-body controllers [9], [10]. Many of these RL methods [2], [3], [8], [11] support teleoperation, enabling demonstration data collection for training a motion planner.

Among those RL methods, the first category is learning from scratch, where the desired behavior is specified through a complex reward function. However, learning loco-manipulation without demonstrations is challenging due to the high dimensionality of humanoid systems and the diversity and compositional complexity of loco-manipulation tasks. Several studies have demonstrated successful sim-to-real transfer for autonomous object manipulation [12]–[15]. These policies are typically tuned for each specific task, making it difficult to acquire long-horizon, autonomous loco-manipulation skills.

The second category uses RL for motion tracking of reference, such as a manually designed trajectory [16], [17] or human data [2], [3], [18]. While this paradigm has achieved versatile motion tracking capability, it fundamentally relies on externally provided references and therefore lacks the task- and environment-level awareness required for true autonomy [2].

In this paper, our loco-manipulation controller falls in the second category. It can walk and manipulate objects with simple commands, including desired base velocity and hand poses. Compared with other whole-body controllers with latent vector inputs [10] or whole-body joint angles [8], our control policy's input space is more physically interpretable and enables more intuitive task-level control, either through VR teleoperation or a high-level planner, such as a diffusion policy [4].

B. Autonomous Humanoid Loco-Manipulation

To achieve autonomous task execution on humanoid robots, many studies adopt modern behavior cloning (BC) techniques, such as DP [4] or action-chunking transformer [19], to learn contact-rich manipulation directly from multi-modal demonstration data. However, while most autonomous policies focus on relatively stable manipulator platforms, extending these approaches to autonomous humanoid loco-manipulation remains challenging due to underactuated and unstable dynamics, and the need to coordinate whole-body motion over long horizons.

Among the few prior works on autonomous humanoid loco-manipulation, some approaches adopt hierarchical frame-

works that decouple high-level planning from low-level whole-body control, thereby keeping the action space simple and interpretable. For example, [3], [8], [20] employs a planner–controller hierarchy to structure decision making and execution. Other approaches maintain a unified end-to-end policy but rely on large-scale domain randomization to integrate perception and improve sim-to-real robustness, as demonstrated in recent work on humanoid door opening [15], [21]. Our REFINE-DP's hierarchical framework falls into the first category, with a high-level planner offering the advantage of learning a diverse range of tasks.

C. Fine-tuning Pre-trained Imitation Learning Policy

To address the inherent issue of compounding error and domain shift in DP, several works have explored integrating RL with DP by learning a residual policy on top of a pre-trained DP to refine actions and enhance performance. These residual policies provide corrective compensations for imitation errors. Existing works have explored on-policy RL [22], [23] and off-policy RL [24] for learning residual refinement to improve precision and sample efficiency.

Another line of work focuses on fine-tuning diffusion policies with RL. In Diffusion Policy Policy Optimization (DPPO) [5], the authors adapt Proximal Policy Optimization (PPO) [25] and fine-tune a pre-trained diffusion policy for a higher success rate. Another work [26] proposes two algorithms, Diffusion Policy Mirror Descent (DPMD) and Soft Diffusion Actor Critic (SDAC), for online training of diffusion policies from scratch. Both of these methods update the diffusion policies directly. More broadly, π_{RL} [27] adopt RL for fine-tuning vision-language-action models. While prior work optimizes only the planner, few works jointly optimize a diffusion-based high-level planner and a low-level controller within a unified planning and control framework.

III. Methods

Our pipeline consists of three stages, as illustrated in Fig. 1. It first collects expert loco-manipulation demonstrations, then pre-trains a diffusion policy (DP), which is subsequently fine-tuned to enhance the task success rate and motion quality.

Collecting loco-manipulation data. Expert demonstrations are collected by teleoperating the robot or by rolling out a heuristic planner in IsaacLab [28]. During the teleoperation, the operator gives commands via a VR device to a pre-trained loco-manipulation policy $\pi_{\text{loco\_manip}}$, as detailed in III-A. In addition to teleoperation, we leverage heuristic planners to scale up data collection. During data collection, a humanoid robot executes diverse loco-manipulation skills, such as object transportation and door opening.

Pre-training the diffusion policy: For each task, we pre-train a DP $\bar{\pi}_\theta$ using the collected dataset. The pre-trained DP outputs base velocity and hand pose commands, which are passed to the low-level loco-manipulation control policy $\pi_{\text{loco\_manip}}$ for execution.

Fine-tuning in target environments: The pre-trained DP is further fine-tuned in the simulator for the same or a more challenging target environment. We adopt two fine-tuning settings: (i) Only the diffusion policy parameters are updated, while the control policy $\pi_{\text{loco\_manip}}$ remains frozen as a low-level controller; (ii) Jointly optimize both the $\bar{\pi}_\theta$ and $\pi_{\text{loco\_manip}}$. Fine-tuning the DP adapts it to domain-specific dynamics by enabling online adaptation beyond offline expert data, substantially improving its success rate. Jointly optimizing both policies not only achieves a high success rate, but also yields better motion quality, namely, more tracking accuracy and less motion jitter.

A. Training Loco-manipulation Controller for Data Collection

We develop a reinforcement learning (RL) policy $\pi_{\text{loco\_manip}}$ capable of simultaneous locomotion and manipulation. The training and deployment pipeline is illustrated in Fig. 2. $\pi_{\text{loco\_manip}}$ adopts a decoupled control architecture that separates upper-body and lower-body behaviors, similar to prior humanoid loco-manipulation works [9]–[11]. Specifically, a lower-body locomotion policy is responsible for achieving intermediate foot placements while maintaining dynamic balance, whereas an upper-body arm policy tracks desired hand poses to execute manipulation tasks.

While most existing locomotion policies are formulated as velocity-tracking controllers, their objectives are designed for long-distance periodic walking rather than for the frequent start–stop transitions and precise torso position adjustments required for manipulation tasks. To enable accurate locomotion toward target positions and achieve a high loco-manipulation success rate, we introduce a foot-placement tracking controller π_{lower} that takes discrete foot-placement commands as input. Unlike velocity-tracking controllers that can accumulate positional error over time, our locomotion policy takes discrete foot-placement commands as input, providing direct control over where each footstep lands. Prior studies [29], [30] have demonstrated that this formulation yields substantially improved foot-placement accuracy and locomotion stability.

We train π_{lower} using RL motion imitation [31] on lower-body reference trajectories [17]. The policy is conditioned on foot-placement commands, which include a swing-foot indicator, a countdown, and a relative target swing-foot pose, alongside proprioception. In addition to the commands, the observation $\mathbf{s}_t^{\text{lower}}$ includes joint states, the gravity vector, and the previous action. The reward function combines foot tracking terms with regularization terms to ensure accuracy and stability. Additionally, we address disturbances from upper-body movement by training with randomized arm configurations.

The goal of the upper-body policy π_{upper} is to accurately track hand pose commands. These commands are SE(3) poses in the robot's root frame. We generate a set of feasible hand commands via sampling collision-free joint configurations and solving forward kinematics. Other than the hand pose command, the observation $\mathbf{s}_t^{\text{upper}}$ includes arm joint states, current hand poses, and tracking errors. To improve robustness to payload and actuation uncertainties, we apply domain randomization during the training of π_{upper}.

Both π_{lower} and π_{upper} output joint position commands $a_t = [a_t^{\text{lower}}; a_t^{\text{upper}}]$, where $a_t^{\text{lower}} \in \mathbb{R}^{12}$ corresponds to the leg joints and $a_t^{\text{upper}} \in \mathbb{R}^{14}$ corresponds to the arm joints. These

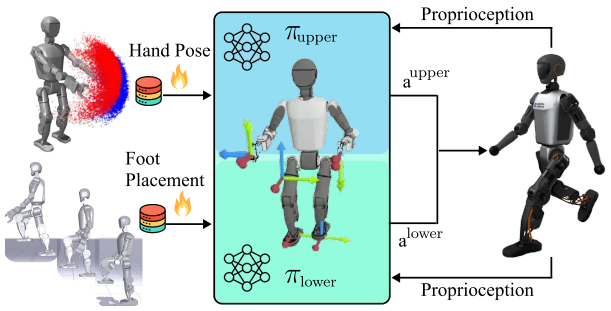

Fig. 2: The sim-to-real RL of the loco-manipulation control policy, from motion prior generation to deployment.

commands specify joint position offsets relative to a fixed default configuration q_{def}. The combined loco-manipulation policy is $\pi_{\text{loco\_manip}}(a_t|s_t)$, where $s_t = [s_t^{\text{lower}}; s_t^{\text{upper}}]$. The resulting target joint positions, $q_{\text{target}} = q_{\text{def}} + a_t$, are tracked by proportional-derivative (PD) control. Detailed observations and rewards are on our website.

The primary methods for data collection include teleoperation in a simulator and the rollout of heuristic planners, both leveraging $\pi_{\text{loco\_manip}}$ for whole-body control. Because teleoperation is costly, we augment 50 teleoperated trajectories with heuristic planner rollouts, expanding the dataset to 1000 trajectories. These heuristic planners are designed for each task via stage-conditioned behavior transitions. To ensure broad coverage of scenarios, we randomize both the initial object configurations and the robot's initial states during data collection. We collect only successful trajectories, each containing state-action pairs. The observed states include the robot's hand and foot poses in the body frame, the gripper state, and object information. The action specifies the desired hand poses, gripper state, and a base-velocity command. We use a velocity-to-footstep planner to convert the velocity command into a sequence of footstep commands.

B. Diffusion Policy Pre-training

The collected loco-manipulation data is used to pre-train a DP [4]. Formally, a DP models a conditional action distribution $p_\theta(\mathbf{A}_t^{0:K} \mid \mathbf{S}_t) = p(\mathbf{A}_t^K)\prod_{k=1}^K p_\theta(\mathbf{A}_t^{k-1} \mid \mathbf{A}_t^k, \mathbf{S}_t)$, where $\mathbf{A}_t^k$ represents an action chunk, and $\mathbf{S}_t$ represents a state observation chunk [19]. The superscript k represents the denoising timestep, and the subscript t represents the environment timestep. The action distribution is learned by modeling a sequence of progressively corrupted actions generated through a forward diffusion process, where Gaussian noise is incrementally added to the actions:

$$q(\mathbf{A}_t^k \mid \mathbf{A}_t^{k-1}) = \mathcal{N}\big(\sqrt{\alpha_k}\mathbf{A}_t^{k-1}, (1-\alpha_k)I\big), \tag{1}$$

where $\{\alpha_k\}_{k=1}^K$ is a pre-scheduled decreasing sequence of coefficients. In practice, we sample noise-corrupted actions directly using a closed-form marginal:

$$\mathbf{A}_t^k = \sqrt{\bar{\alpha}_k}\,\mathbf{A}_t^0 + \sqrt{1-\bar{\alpha}_k}\,\epsilon,$$

where $\bar{\alpha}_k = \prod_{i=1}^k \alpha_i$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, the standard normal distribution. Given the dataset $\mathcal{D} = \{(\mathbf{S}_i, \mathbf{A}_i^0)\}_{i=1}^N$, we maximize the log likelihood, which can be reformulated as training a noise prediction network ϵ_θ by minimizing the noise prediction error [32]:

$$\mathcal{L}_{\text{diff}}(\theta) = \mathbb{E}_{\mathbf{A}_t^0, \mathbf{S}_t, k, \epsilon}\Big[\|\epsilon - \epsilon_\theta(\mathbf{A}_t^k, \mathbf{S}_t, k)\|^2\Big]. \tag{2}$$

The action generation involves a reverse denoising process. At inference time, we samples an initial Gaussian noise $\mathbf{A}_t^K \sim \mathcal{N}(0, \mathbf{I})$ and progressively denoise it to produce the clean action $\mathbf{A}_t^0$. At each denoising step, the policy ϵ_θ predicts the noise component that refines a noisy action sample $\mathbf{A}_t^k$ towards the next action $\mathbf{A}_t^{k-1}$:

$$p_\theta(\mathbf{A}_t^{k-1} \mid \mathbf{A}_t^k, \mathbf{S}_t) = \mathcal{N}(\mathbf{A}_t^{k-1}; \boldsymbol{\mu}_\theta(\mathbf{A}_t^k, \mathbf{S}_t), \boldsymbol{\sigma}_k) \tag{3}$$

where $\boldsymbol{\mu}_\theta(\mathbf{A}_t^k, \mathbf{S}_t) = \frac{1}{\sqrt{\alpha_k}}\Big(\mathbf{A}_t^k - \frac{(1-\alpha_k)}{\sqrt{1-\bar{\alpha}_k}}\epsilon_\theta(\mathbf{A}_t^k, \mathbf{S}_t, k)\Big)$, and $\boldsymbol{\sigma}_k = \frac{1-\bar{\alpha}_{k-1}}{1-\bar{\alpha}_k}\cdot(1-\alpha_k)$.

C. Diffusion Policy Fine-tuning

While using the DP as a high-level planner achieves a decent success rate, its performance remains insufficient for reliable task execution. In particular, the pre-trained planner does not explicitly account for the robot's closed-loop dynamics and execution errors, resulting in deviations that accumulate over long horizons and occasional task failure. This compounding error in pre-training motivates us to further enhance the policy. In this study, we employ RL to fine-tune the pre-trained DP. RL fine-tuning offers two key advantages: (i) improving task success through trial-and-error interaction with the simulation environment, and (ii) enhancing robustness to out-of-distribution conditions through domain randomization, enabling zero-shot sim-to-real transfer.

Conventional policy gradient methods such as Proximal Policy Optimization (PPO) [25] rely on evaluating a policy density function $\bar{\pi}_\theta(\mathbf{A}_t^0|\mathbf{S}_t)$. However, DPs are implicit policies, for which $\bar{\pi}_\theta(\mathbf{A}_t^0|\mathbf{S}_t)$ is not tractable. To address this, we adopt Diffusion Policy Policy Optimization (DPPO) [5] that augments an environment Markov Decision Process (MDP) by incorporating the denoising process into the MDP and treating each denoising step as a decision step. Since denoising transitions follow a tractable Gaussian distribution, this augmented MDP enables a likelihood-based policy gradient method, such as PPO [25], allowing RL fine-tuning of DP.

Formally, we define the environment MDP as $\mathcal{M}_{\text{ENV}} := (\mathcal{S}, \mathcal{A}, P_0, P, R)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P_0 is the initial state distribution, P is the transition probabilities, and R is the reward function. We then define a diffusion-process-augmented MDP $\mathcal{M}_{\text{DP}} := (\bar{\mathcal{S}}, \bar{\mathcal{A}}, \bar{P}_0, \bar{P}, \bar{R})$, which expands $\mathcal{M}_{\text{ENV}}$ by inserting one full denoising process in each environment timestep. This augmented $\mathcal{M}_{\text{DP}}$ defines a unified timestep index $\bar{t}(t,k) = tK + (K-1-k)$, where K is the total number of denoising steps and $k \in [0, K-1]$. Accordingly, the augmented state, action, and initial state distribution are given by

$$\bar{s}_{\bar{t}(t,k)} = (\mathbf{S}_t, \mathbf{A}_t^{k+1}), \quad \bar{a}_{\bar{t}(t,k)} = \mathbf{A}_t^k, \quad \bar{P}_0 = P_0 \otimes \mathcal{N}(0, I).$$

For $k > 0$, the $\mathcal{M}_{\text{DP}}$ propagates the denoising process; the policy outputs noisy action samples $\mathbf{A}_t^k$, which are not executed in the environment. Consequently, these intermediate steps receive zero reward. Only the denoised action $\mathbf{A}_t^0$ at the final

denoising step ($k = 0$) is executed in the environment MDP $\mathcal{M}_{\text{ENV}}$, where it induces a reward through the environment dynamics. Accordingly, the transition dynamics are defined as

$$\bar{P}(\bar{s}_{\bar{t}+1} \mid \bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) = \begin{cases} (\mathbf{S}_t, \mathbf{A}_t^k) \sim \boldsymbol{\delta}_{\mathbf{S}_t, \mathbf{A}_t^k}, & k > 0 \\ (\mathbf{S}_{t+1}, \mathbf{A}_{t+1}^K) \sim P(\mathbf{S}_{t+1} \mid \mathbf{S}_t, \mathbf{A}_t^0) \otimes \mathcal{N}(0, \mathbf{I}), & k = 0 \end{cases} \quad (4)$$

where $\boldsymbol{\delta}$ is the Dirac distribution. The corresponding reward function used for fine-tuning the diffusion policy is given by

$$\bar{R}_{\bar{t}(t,k)}\left(\bar{s}_{\bar{t}(t,k)}, \bar{a}_{\bar{t}(t,k)}\right) = \begin{cases} 0, & k > 0 \\ R_t(\mathbf{S}_t, \mathbf{A}_t^0), & k = 0 \end{cases} \quad (5)$$

For the DP update, we optimize the objective:

$$\nabla_\theta \bar{\mathcal{J}}(\bar{\pi}_\theta) = \mathbb{E}_{\bar{\pi}_\theta, \bar{P}, \bar{P}_0}\left[\sum_{\bar{t} \geq 0} \nabla_\theta \log \bar{\pi}_\theta(\bar{a}_{\bar{t}} \mid \bar{s}_{\bar{t}})\, \bar{r}(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}})\right],$$
$$\bar{r}(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}}) := \sum_{\tau \geq \bar{t}} \gamma(\tau)\, \bar{R}_\tau(\bar{s}_\tau, \bar{a}_\tau), \quad (6)$$

where $\bar{\pi}_\theta(\bar{a}_{\bar{t}(t,k)} \mid \bar{s}_{\bar{t}(t,k)}) = p_\theta(\mathbf{A}_t^k | \mathbf{A}_t^{k+1}, \mathbf{S}_t)$ denotes a conditional distribution of the action $\mathbf{A}_t^k$, and we optimize the log likelihood of the action sequence $\mathbf{A}_t^{K:0}$. We use generalized advantage estimation (GAE) [33] to calculate the advantage estimate $\hat{A}(\bar{s}_{\bar{t}}, \bar{a}_{\bar{t}})$ from a replay buffer and optimize the DP using a PPO-style policy gradient.

D. Joint Optimization of the Diffusion and the RL Policies

To improve the motion quality beyond the success rate, we propose a joint optimization technique that fine-tunes both the high-level DP and the low-level RL controller. Unlike the fine-tuning described in Sec. III-C that optimizes only the DP, we additionally optimize the low-level RL control policy.

As shown in Fig. 1(c), the augmented environment $\mathcal{M}_{\text{DP}}$ produces two sets of states $(\bar{s}_t, s_t)$ and rewards $(\bar{R}_t, \hat{R}_t)$. $\bar{R}_t$ is used to update the DP $\bar{\pi}_\theta(\bar{a}_t \mid \bar{s}_t)$ and enhance task success rate. $\hat{R}_t$ represents a set of rewards used in pre-training the low-level control policy $\pi_{\text{loco\_manip}}$, and its purpose is to further improve motion quality and enable accurate and smooth tracking of the DP commands. Both the DPs and the RL control policy are optimized using PPO. A detailed description of the algorithm is in Alg. 1.

Joint optimization enables the tracking controller to accurately follow dynamic, task-relevant commands generated by the DP planner. Unlike the independently sampled, stationary commands used during controller pre-training, DP commands represent a moving target along a continuous trajectory, creating a distribution mismatch that degrades tracking performance. Joint optimization alleviates this mismatch by exposing the controller to planner-generated commands, bringing them in-distribution and substantially improving tracking accuracy. Combined with the optimization of DP, this process enhances low-level motion quality and achieves a high success rate.

Algorithm 1 REFINE-DP: Joint Optimization of DP and RL

Input: Pre-trained diffusion policy $\bar{\pi}_\theta(\mathbf{A}_t^0 | \mathbf{S}_t)$, low-level control policy $\pi_{\text{loco\_manip}}$, augmented MDP $\mathcal{M}_{\text{DP}}$, replay buffer $D = \bar{D} = \emptyset$.
Initialize state $\bar{s}_t$ from $\mathcal{M}_{\text{DP}}$.
for $iter = 0$ **to** L **do**
  for $iter = 0$ **to** M **do**
    Sample an action chunk from DP $\mathbf{A}_t^0 = \bar{\pi}_\theta(\mathbf{A}_t^0 | \mathbf{S}_t)$.
    Execution $(\bar{s}_t, \bar{a}_t, \bar{R}_t), (s_t, a_t, \hat{R}_t) = \mathcal{M}_{\text{DP}}(\mathbf{A}_t^0)$.
    $D \leftarrow D \cup \{D'\}$, $D' = (s_t, a_t, \hat{R}_t)$.
    Sample a mini-batch D_k from D.
    Update $\pi_{\text{loco\_manip}}$ using D_k via PPO [25].
  end for
  for $iter = 0$ **to** N **do**
    Sample an action chunk from DP $\mathbf{A}_t^0 = \bar{\pi}_\theta(\mathbf{A}_t^0 | \mathbf{S}_t)$.
    Execution $(\bar{s}_t, \bar{a}_t, \bar{R}_t), (s_t, a_t, \hat{R}_t) = \mathcal{M}_{\text{DP}}(\mathbf{A}_t^0)$.
    $\bar{D} \leftarrow \bar{D} \cup \{D'\}$, $D' = (\bar{s}_t, \bar{a}_t, \bar{R}_t)$.
    Sample a mini-batch $\bar{D}_k$ from $\bar{D}$.
    Update $\bar{\pi}_\theta(\mathbf{A}_t^0 | \mathbf{S}_t)$ using $\bar{D}_k$ via DPPO (6).
  end for
end for

IV. Experiments

A. Experiment Setup

For our hardware experiments, we use a Booster T1 robot with 29 degrees of freedom. Both the high-level diffusion policy (DP) $\bar{\pi}_\theta$ and the low-level reinforcement learning (RL) controller $\pi_{\text{loco\_manip}}$ run on a policy computer, with an AMD 7945HX CPU and an NVIDIA RTX 4060 GPU. This policy computer receives the robot's proprioception states (i.e., joint angles and IMU measurements) from T1's embedded computer. The policy computer also performs the inference of $\bar{\pi}_\theta$ that outputs a high-level command for the $\pi_{\text{loco\_manip}}$, which then outputs joint position commands. We use a state-based DP that relies on the pre-processed pose of both the robot torso and the target object, which are captured using a motion capture (MoCap) system. A MoCap workstation streams these object-state observations to the policy computer at 90 Hz.

The DP $\bar{\pi}_\theta$ has an observation horizon of 8 and an action chunk size of 12, with a 0.1 s interval between each observation and action. We choose $\bar{\alpha}_k$ to follow the cosine schedule [34] for DP pre-training. The DP $\bar{\pi}_\theta$ runs at 10 Hz using NVIDIA TensorRT. The loco-manipulation policy $\pi_{\text{loco\_manip}}$ runs at 50 Hz. For sim-to-real RL policy training, expert demonstration collection, and fine-tuning, we use IsaacLab [28] as the simulator.

As shown in Fig. 3, we evaluate our REFINE-DP on a set of loco-manipulation tasks. Each of these tasks requires multiple stages of walking and manipulation. Unlike [35] that requires a stage signal, our DP policy infers task stages from object locations and implicitly learns stage transitions, enabling loco-manipulation without explicit stage supervision.

Task 1: Object pickup; Task 2: Long-horizon pick-and-place. Task 1 requires walking to a table to pick up a box, whereas Task 2 is a longer-horizon extension (i.e., 40 s) with placing the object on another table.

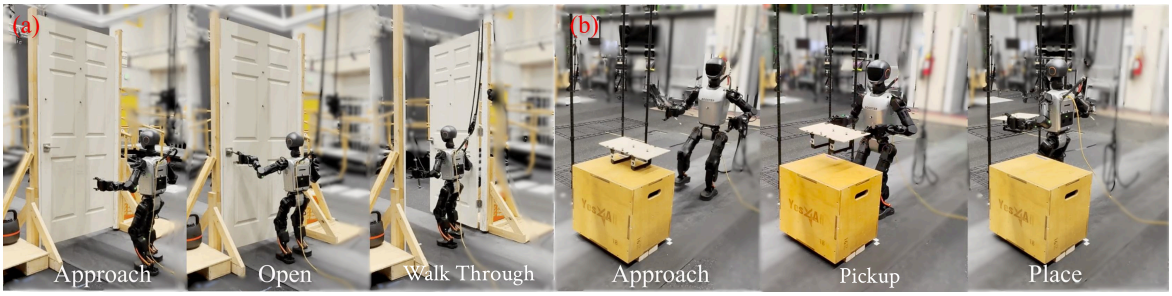

Fig. 3: The robot autonomously executes loco-manipulation tasks: (a) door opening, (b) object transportation.

Task 3: Door opening and traversal. For humanoid door-opening tasks, prior work relies on heuristic planners [36], sim-to-real RL [15], or action-chunking transformer (ACT) [35]. In contrast, we are the first to apply DP to humanoid door traversal.

Task 4: Stair-assisted object retrieval. This loco-manipulation task involves uneven terrain, where the humanoid must step onto an elevated platform to retrieve an object.

B. Baselines Methods and Ablation Study

We compare REFINE-DP with several baselines that use different planner architectures or fine-tuning strategies. Specifically, we consider two ablations: (i) a pre-trained DP with transformer backbone (DiT), (ii) a deterministic multilayer perceptron (MLP) backbone; and two baselines: (iii) fine-tuning the pre-trained MLP backbone (MLP-FT), and (iv) residual RL [22] to further finetune the diffusion policy. These experiments benchmark the high-level planner, with the low-level $\pi_{\text{loco\_manip}}$ remaining unchanged.

DiT. The DP baseline follows the standard formulation [4] using a transformer backbone. This evaluates the performance of a purely pre-trained DP within our hierarchical framework.

MLP. The MLP baseline replaces the transformer backbone from DiT with an MLP. Unlike DPs, the MLP does not model uncertainty or multi-modality in the action distribution and therefore produces deterministic outputs at inference time.

MLP-FT. MLP-FT extends the deterministic MLP baseline by fine-tuning via RL, serving as an ablation of the planner backbone. Since policy fine-tuning relies on stochastic exploration, we convert the pre-trained deterministic MLP into a stochastic policy during fine-tuning by interpreting its output as the mean and sampling around it. Specifically, during fine-tuning, we sample action through a stochastic process with a mean-reverting update:

$$\mathbf{A}_t^{k-1} = (1-\lambda_k)\mathbf{A}_t^k + \lambda_k \tilde{\boldsymbol{\mu}}_\theta(\mathbf{S}_t) + \tilde{\boldsymbol{\sigma}}_k \boldsymbol{\epsilon}_k, \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \mathbf{I}),$$

where $\tilde{\boldsymbol{\mu}}_\theta(\mathbf{S}_t)$ denotes the MLP mean action prediction. The interpolation coefficient λ_k pulls samples toward the mean policy, while $\tilde{\boldsymbol{\sigma}}_k$ controls the exploration magnitude. We adopt a linear schedule for λ_k and a decreasing noise schedule for $\tilde{\boldsymbol{\sigma}}_k$, encouraging exploration in early steps and concentration around the mean policy in later steps.

This design enables RL fine-tuning of the MLP baseline while preserving its deterministic structure, allowing us to isolate the contribution of the diffusion-based policy formulation from the fine-tuning procedure itself.

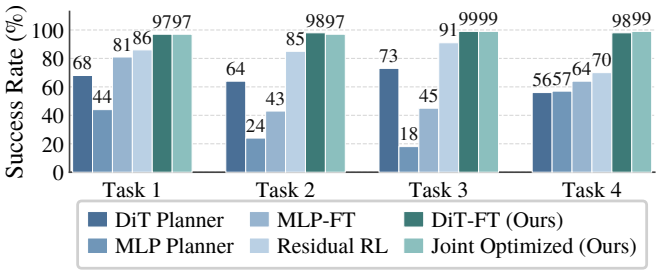

Fig. 4: Comparison of success rates across tasks in Sec. IV-A and methods in Sec. IV-B.

Residual RL. Residual RL [22] improves a pre-trained DP by learning a lightweight corrective residual that is added to the output of the frozen DP. Rather than modifying the parameters of a pre-trained DP, a residual Gaussian policy is optimized with PPO to predict small additive adjustments to the frozen DP's actions.

C. Quantitative Results and Analysis

C-1 Task Success and Planner Capacity

REFINE-DP improves task success rate (SR). The task SR comparison with the baselines is reported in Fig. 4. The results show that REFINE-DP, including fine-tuning and joint optimization, substantially outperforms all baselines across all loco-manipulation tasks. REFINE-DP can achieve more than 90% by fine-tuning from the pre-trained policy of $50-70\%$. Since the fine-tuned SR is already high, joint optimization does not further improve SR; instead, it improves motion quality.

Transformer backbone shows a superior SR than MLP. We observe a consistent performance gain in the transformer-based diffusion (DiT) over the MLP-based diffusion in Fig. 4. We attribute this improvement to DiT's ability to model multi-modal action distributions and generate diverse feasible trajectories. In contrast, MLPs perform deterministic regression and tend to average across demonstrations. When multiple valid solutions exist, this averaging effect can produce actions that may not correspond to a viable trajectory.

MLP-FT partially mitigates this limitation by injecting stochasticity during fine-tuning, enabling exploration and recovery from averaged behaviors. Performance improves as the policy adapts to the target environment and reduces the distribution gap between offline demonstrations and online execution. However, the achievable improvement remains limited by the representational capacity of the backbone. These results

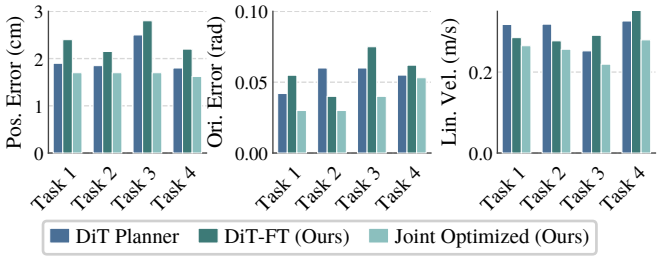

Fig. 5: Joint optimization achieves better end effector tracking performances and velocities across tasks in Sec. IV-A.

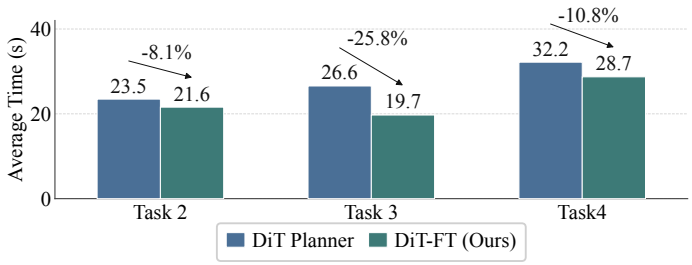

Fig. 6: Fine-tuning reduces task execution time in both simulation and hardware experiments.

highlight the importance of backbone capacity for multi-modal action modeling and the effectiveness of fine-tuning.

Pre-training is essential for successful fine-tuning. Successful fine-tuning requires a sufficiently capable pre-trained policy. We find that fine-tuning is effective when a pre-trained policy achieves a moderate success rate (approximately $50-70\%$). In contrast, fine-tuning from random initialization shows limited improvement, as sparse rewards rarely provide informative learning signals when task success is infrequent. Pre-training alleviates the need for extensive reward shaping typically required in RL from scratch [15]. This allows the policy to improve its SR with minimal manual tuning.

C-2 Efficiency Gains from Fine-tuning and Joint Optimization

Joint optimization improves fine-tuning efficiency and motion quality. Optimizing the low-level controller provides three benefits. First, optimizing the RL controller alone using pre-trained DP rollouts D already improves the SR by 18% on the long-horizon pick-and-place task. Second, the jointly optimized low-level policy $\pi'_{\text{loco\_manip}}$ improves the training efficiency of DP fine-tuning, requiring approximately half as many iterations (20 instead of 40) to achieve a 90% SR compared to fine-tuning with the pre-trained low-level RL policy $\pi_{\text{loco\_manip}}$. Third, joint optimization improves motion quality in addition to a consistently high SR, as shown in Fig. 4. Specifically, we evaluate upper-body tracking performance using position and orientation errors, as well as motion smoothness using linear velocity, averaged across 100 trials. Collectively, $\pi'_{\text{loco\_manip}}$ achieves the lowest position error and reduces orientation error by up to 50% compared to the pre-trained $\pi_{\text{loco\_manip}}$, as shown in Fig. 5. This strict lower tracking error coincides with an average of 15% decrease in end-effector velocity, reflecting smoother manipulation. Meanwhile, fine-tuning the DP alone can degrade low-level tracking performance, as it adapts to poor tracking accuracy and learns to overcommand for task success.

Fine-tuning improves data efficiency. We further evaluate the data efficiency of fine-tuning by measuring the SR as a function of the offline dataset size. Fig. 7 compares the data scaling behavior with and without fine-tuning on both the box-pick-and-place and door-opening tasks. While the SR improves with more pre-training data, fine-tuning achieves higher performance with substantially fewer demonstrations. In particular, reaching a 90% SR through pre-training alone requires approximately $1,000$ trajectories. In contrast, a DP pre-trained on only 50 trajectories achieves up to 95% SR after fine-tuning. These results demonstrate that fine-tuning significantly reduces the need for expensive pre-training data while yielding a higher final SR.

Fine-tuning achieves higher throughput. The fine-tuned policy demonstrates higher throughput, completing the same task in less time. As shown in Fig. 6, fine-tuning reduces task completion time by 15% on average across Task 2, Task 3, and Task 4, due to the improved action efficiency acquired through RL. In contrast, the pre-trained policy often produces hesitant or indecisive motions, resulting in slower execution.

Fine-tuning adapts to out-of-distribution (OOD) scenarios in pre-training. To extend the pre-trained policy to OOD scenarios, we employ domain randomization to expose it to scenarios beyond those observed during pre-training. Specifically, we fine-tune for the object transport task in unseen robot initial locations, which are parameterized by three variables relative to the target object: (i) radial distance, (ii) polar angle, and (iii) heading angle of the robot. The fine-tuning expands the ranges to cover 320%, 125%, and 600% of the corresponding pre-training ranges, respectively. Initially, the policy struggles in the OOD scenarios and rarely receives the sparse task-success reward, leading to inefficient RL fine-tuning. To improve the fine-tuning efficiency, we introduce a curriculum that progressively increases the ranges by 10% once the policy achieves 90% SR at the current randomization levels. This curriculum enables the policy to gradually expand coverage and master increasingly challenging conditions. While the pre-trained policy achieves only 7% SR at the maximum randomization level, our curriculum-based fine-tuning improves the SR to over 90%. Fine-tuning directly on maximum randomization without a curriculum collapses early and achieves only 30% terminal SR.

D. Hardware Experiment

We demonstrate the autonomous loco-manipulation capability of our framework on a Booster T1 humanoid robot, as shown in Fig. 3.

For stable and accurate execution, we clamp locomotion speed to 0.2 m/s and hand speed to 0.05 m/s. This enables accurate loco-manipulation, leading to higher SR. In real-world experiments, REFINE-DP exceeds 70% SR in the door-opening task and 50% SR in the box transportation task.

In hardware experiments, the sim-to-real gap arises from two primary sources. First, the object observations, such as door handle positions and target box poses, are noisy. They exhibit offsets compared to the scene in simulation and can

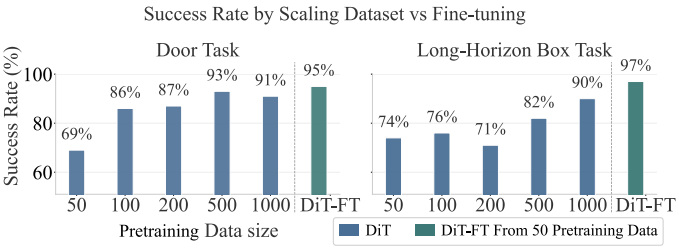

Fig. 7: Comparison of success rate between scaling pre-training data and fine-tuning diffusion policy.

become highly noisy when the robot is in motion. Second, locomotion exhibits more trembling and sliding behaviors. These factors lead to a reduction from the $90\%+$ SR in simulation. Despite these gaps, the fine-tuned policy demonstrates improved robustness to noisy observations and execution errors.

REFINE-DP adapts to environmental perturbations during execution by re-adjusting and re-attempting. As demonstrated in Task 1, when the target object location is altered, the robot re-adjusts its plan on-the-fly and continues the task. Upon execution failures, the policy re-attempts the task. For example, in the door-opening experiment, when the robot misses the door handle, it often takes small footsteps to get closer before attempting the manipulation again. Such corrective behaviors occur consistently under the fine-tuned policy, whereas the pre-trained policy often becomes stuck and fails to recover, leading to halted execution. This capability to re-attempt and recover from unexpected changes and failed actions demonstrates the robustness and autonomy of our REFINE-DP.

During fine-tuning, REFINE-DP improves task efficiency by eliminating redundant and indecisive motions commonly observed in pre-trained policies, thereby increasing task throughput. REFINE-DP achieves average speedups of around 10% and 20% for the box pickup and door-opening tasks.

V. Conclusion

In this study, we introduced REFINE-DP, a fine-tuning framework that jointly optimizes a hierarchical planner–controller architecture. Fine-tuning the high-level diffusion-policy planner improves data efficiency and enhances generalization to out-of-distribution scenarios. Meanwhile, fine-tuning the low-level loco-manipulation controller improves motion tracking quality, yielding higher tracking accuracy and smoother execution. Together, these components achieve high success rates on long-horizon loco-manipulation tasks on a humanoid robot. Future work will focus on incorporating vision modality in the planner and allowing more versatile whole-body skills in the controller.

References

[1] C. Khazoom, S. Hong, M. Chignoli, E. Stanger-Jones, and S. Kim, "Tailoring solution accuracy for fast whole-body model predictive control of legged robots," *IEEE Robotics and Automation Letters*, vol. 9, no. 12, pp. 11 074–11 081, 2024.

[2] T. He *et al.*, "Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning." in *CoRL*, vol. 270, 2024, pp. 1516–1540.

[3] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," in *Annual Conference on Robot Learning*, 2024.

[4] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 44, no. 10-11, pp. 1684–1704, 2025.

[5] A. Z. Ren *et al.*, "Diffusion policy policy optimization," in *The Thirteenth International Conference on Learning Representations*, 2025.

[6] H. Jung, Z. Gu, Y. Zhao, H.-W. Park, and S. Ha, "Ppf: Pre-training and preservative fine-tuning of humanoid locomotion via model-assumption-based regularization," *IEEE Robotics and Automation Letters*, vol. 10, no. 11, pp. 11 466–11 473, 2025.

[7] Z. Gu *et al.*, "Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning," *IEEE/ASME Transactions on Mechatronics. DOI: 10.1109/TMECH.2025.3579247*, 2025.

[8] Y. Ze *et al.*, "Twist2: Scalable, portable, and holistic humanoid data collection system," *arXiv preprint arXiv:2511.02832*, 2025.

[9] Y. Zhang *et al.*, "Falcon: Learning force-adaptive humanoid loco-manipulation," 2025.

[10] C. Lu *et al.*, "Mobile-television: Predictive motion priors for humanoid whole-body control," *arXiv preprint arXiv:2412.07773*, 2024.

[11] Q. Ben, F. Jia, J. Zeng, J. Dong, D. Lin, and J. Pang, "HOMIE: Humanoid Loco-Manipulation with Isomorphic Exoskeleton Cockpit," in *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025.

[12] C. Zhang, W. Xiao, T. He, and G. Shi, "Wococo: Learning whole-body humanoid control with sequential contacts," in *Annual Conference on Robot Learning*, 2024.

[13] C. Schwarke, V. Klemm, M. v. d. Boon, M. Bjelonic, and M. Hutter, "Curiosity-driven learning of joint locomotion and manipulation tasks," in *Proceedings of the 7th Conference on Robot Learning*, vol. 229, 2023, pp. 2594–2610.

[14] T. Haarnoja *et al.*, "Learning agile soccer skills for a bipedal robot with deep reinforcement learning," *Science Robotics*, vol. 9, no. 89, 2024.

[15] H. Xue *et al.*, "Opening the sim-to-real door for humanoid pixel-to-action policy transfer," 2025.

[16] J. Dao, H. Duan, and A. Fern, "Sim-to-real learning for humanoid box loco-manipulation," in *International Conference on Robotics and Automation*, 2023.

[17] F. Liu *et al.*, "Opt2skill: Imitating dynamically-feasible whole-body trajectories for versatile humanoid loco-manipulation," *IEEE Robotics and Automation Letters*, vol. 10, no. 11, pp. 12 261–12 268, 2025.

[18] Z. Xie, J. Tseng, S. Starke, M. van de Panne, and C. K. Liu, "Hierarchical planning and control for box loco-manipulation," *Proc. ACM Comput. Graph. Interact. Tech.*, vol. 6, no. 3, 2023.

[19] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[20] S. Yin, Y. Ze, H.-X. Yu, C. K. Liu, and J. Wu, "Visualmimic: Visual humanoid loco-manipulation via motion tracking and generation," 2025.

[21] H. Weng, Y. Li, N. Sobanbabu, Z. Wang, Z. Luo, T. He, D. Ramanan, and G. Shi, "Hdmi: Learning interactive humanoid whole-body control from human videos," 2025.

[22] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal, "From imitation to refinement – residual rl for precise assembly," 2024.

[23] E. Su, T. Westenbroek, A. Nagabandi, and A. Gupta, "Rfs: Reinforcement learning with residual flow steering for dexterous manipulation," 2026.

[24] L. Ankile, Z. Jiang, R. Duan, G. Shi, P. Abbeel, and A. Nagabandi, "Residual off-policy rl for finetuning behavior cloning policies," 2025.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *Computing Research Repository*, vol. abs/1707.06347, 2017.

[26] H. Ma, T. Chen, K. Wang, N. Li, and B. Dai, "Efficient online reinforcement learning for diffusion policy," 2025.

[27] K. Chen *et al.*, "π_{RL}: Online rl fine-tuning for flow-based vision-language-action models," 2026.

[28] M. Mittal *et al.*, "Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning," *arXiv preprint arXiv:2511.04831*, 2025.

[29] H. Wang, Z. Wang, J. Ren, Q. Ben, T. Huang, W. Zhang, and J. Pang, "Beamdojo: Learning agile humanoid locomotion on sparse footholds," *arXiv preprint arXiv:2502.10363*, 2025.

[30] W. Suliman, E. Davydenko, E. Chaikovskaia, and R. Gorbachev, "Reinforcement learning-based footstep control for humanoid robots on complex terrain," *IEEE Access*, 2025.

[31] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deepmimic: example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions On Graphics*, vol. 37, no. 4, 2018.

[32] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
[33] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *International Conference on Learning Representations*, 2016.
[34] A. Q. Nichol and P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 8162–8171.
[35] M. Lee, D. K. Kim, J. K. Bandi, M. Smith, A. Liao, A. akbar Aghamohammadi, and S. Omidshafiei, "Stageact: Stage-conditioned imitation for robust humanoid door opening," 2025.
[36] D. Calvert *et al.*, "A behavior architecture for fast humanoid robot door traversals," *Robotics and Autonomous Systems*, vol. 195, p. 105217, 2026.

VI. APPENDIX

TABLE I: Observation terms for RL Policy training

Observation	**Actor**	**Critic**
Manipulation		
Projected gravity	✓	✓
Arm joint pos.	✓	✓
Arm joint vel.	✓	✓
Action	✓	✓
Hand pose cmd. $(\mathbf{p}_L^{\text{cmd}}, \mathbf{p}_R^{\text{cmd}})$	✓	✓
Current hand poses $(\mathbf{p}_L, \mathbf{p}_R)$	✓	✓
Hand tracking error $(\mathbf{e}_L, \mathbf{e}_R)$	✓	✓
Locomotion		
Foot placement cmd.	✓	✓
Base angular vel.	✓	✓
Projected gravity	✓	✓
Leg joint pos.	✓	✓
Leg joint vel.	✓	✓
Action	✓	✓
Reference foot pos.		✓
Current foot pos.		✓
Base linear vel.		✓
Foot wrench		✓

TABLE II: Rewards terms for foot placement tracking Policy

Term	**Formulation**	**Weight**
End-Effector Reference Tracking (per foot)		
Position ($\sigma = 0.05$)	$\exp\left(-\frac{\|\mathbf{p}-\mathbf{p}^{\text{ref}}\|^2}{\sigma^2}\right)$	$+5.0$
Yaw ($\sigma = 0.1$)	$\exp\left(-\frac{(\psi-\psi^{\text{ref}})^2}{\sigma^2}\right)$	$+3.0$
Base linear vel. ($\sigma = 0.25$)	$\exp\left(-\frac{\|\mathbf{v}-\mathbf{v}^{\text{ref}}\|^2}{\sigma^2}\right)$	$+1.0$
Joint Reference Tracking		
Position ($\sigma = 0.3$)	$\exp\left(-\frac{\|\mathbf{q}-\mathbf{q}^{\text{ref}}\|^2}{\sigma^2}\right)$	$+0.5$
Velocity ($\sigma = 1.0$)	$\exp\left(-\frac{\|\dot{\mathbf{q}}-\dot{\mathbf{q}}^{\text{ref}}\|^2}{\sigma^2}\right)$	$+0.3$
Regularization (with curriculum $\rightarrow$)		
Action smoothness	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$	$-0.001 \rightarrow -0.5$
Joint acceleration	$\|\ddot{\mathbf{q}}\|^2$	$-10^{-7} \rightarrow -10^{-5}$
Torque limits	$\max(0, \lvert\boldsymbol{\tau}\rvert - \boldsymbol{\tau}_{\max})$	-0.5
Joint limits	$\max(0, \lvert\mathbf{q}\rvert - \mathbf{q}_{\max})$	-10^{-4}
Ankle posture	$\|\mathbf{q} - \mathbf{q}_{\text{def}}\|^2$	-2.0
Base ort. ($\sigma = 0.2$)	$\exp\left(-\frac{\|\mathbf{g}_{\text{xy}}\|^2}{\sigma^2}\right)$	$+3.0$
Feet airtime ($\tau = 0.4$)	$\sum_i c_i\left(t_i^{\text{air}} - \tau\right)$	$+20.0$
Base height range	$\left[\max(0,\, h_{\min} - h)\right]^2 + \left[\max(0,\, h - h_{\max})\right]^2$	-3.0

TABLE III: Reward terms for hand pose tracking Policy

Term	**Formulation**	**Weight**
End-Effector Position Tracking (per hand)		
Coarse ($\sigma = 0.4$)	$\exp(-\|\mathbf{p} - \mathbf{p}^{\text{cmd}}\|^2/\sigma^2)$	$+2.0$
Fine ($\sigma = 0.1$)	$\exp(-\|\mathbf{p} - \mathbf{p}^{\text{cmd}}\|^2/\sigma^2)$	$+2.0$
Precise ($\sigma = 0.08$)	$\exp(-\|\mathbf{p} - \mathbf{p}^{\text{cmd}}\|^2/\sigma^2)$	$+2.0$
End-Effector Orientation Tracking (per hand)		
Coarse ($\sigma = 0.8$)	$\exp(-\|\mathbf{e}_{\text{quat}}\|^2/\sigma^2)$	$+1.0$
Fine ($\sigma = 0.5$)	$\exp(-\|\mathbf{e}_{\text{quat}}\|^2/\sigma^2)$	$+1.0$
Precise ($\sigma = 0.3$)	$\exp(-\|\mathbf{e}_{\text{quat}}\|^2/\sigma^2)$	$+1.0$
Regularization (with curriculum $\rightarrow$)		
Posture prior	$\|\mathbf{q} - \mathbf{q}_{\text{nom}}\|_1$	-0.2
Action smoothness	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$	$-0.01 \rightarrow -0.1$
Joint velocity	$\|\dot{\mathbf{q}}\|^2$	$-10^{-3} \rightarrow -2 * 10^{-3}$
Joint acceleration	$\|\ddot{\mathbf{q}}\|^2$	$-10^{-6} \rightarrow -3 * 10^{-6}$
EE acceleration	$\|\ddot{\mathbf{p}}_{\text{ee}}\|^2$	-1.4×10^{-2}
Torque limits	$\max(0, \|\boldsymbol{\tau}\| - \boldsymbol{\tau}_{\max})$	-0.1
Joint limits	$\max(0, \|\mathbf{q}\| - \mathbf{q}_{\max})$	-4.0