

FetchMan: Learning Visual Humanoid Loco-Manipulation Policies from Simulated Experiences

Omar Rayyan^{1*} Zhi Li¹ Max Argus^{2,3} Yuxin Jiang¹
 Chang Yu¹ Chenfanfu Jiang¹ Yuchen Cui¹

¹University of California, Los Angeles ²Allen Institute for AI ³University of Washington

orayyan.com/fetchman

Abstract: Visual loco-manipulation policies that can generalize to novel scenes and objects have long been a goal of robotics research. However, today’s data-hungry algorithms make collecting sufficient demonstrations a struggle for tabletop manipulation, and even more so for humanoids that must also walk and balance. Learning from simulated data and transferring that behavior to the real world, as is commonly done in locomotion, sidesteps this struggle, so we replicate that recipe for loco-manipulation. In doing so, we find that cloning synthetic demonstrations results in a low performance ceiling no matter the amount of training data. Reinforcement learning breaks through it, and refining the cloned policy with Flow-GRPO on a single sparse reward yields performance that synthetic behavior cloning cannot match. Together, these stages form our end-to-end sim-to-real pipeline spanning more than 150,000 scenes, which we use to train **FETCHMAN**. We evaluate it on **FETCHMAN-BENCH**, a simulation benchmark we release, and deploy it zero-shot on a real Unitree G1, where our single-object reach-and-pick policy walks to and grasps a target across unseen scenes at 73.3% success. Finally, we extend this recipe to multi-object training, a first step toward loco-manipulation generalist policies at this data scale.



Figure 1: **FETCHMAN** learns visual humanoid loco-manipulation entirely in simulation and deploys it zero-shot to a real Unitree G1. The resulting policy walks up to and grasps objects across diverse, unseen real-world scenes, with no real-world data or per-scene tuning.

*Correspondence to orayyan@ucla.edu

1 Introduction

Simulation has become the standard setting for training humanoid locomotion policies, where interaction with the environment is typically limited to feet-floor contact and observations are low-dimensional. With easy-to-use tools [1, 2, 3], a walking, dancing, or back-flipping policy can be brought to life. This success has yet to translate to loco-manipulation, for two reasons. Diverse simulation assets are scarce and take significant engineering effort to scale, and contact-rich interaction is harder to model than the relatively self-contained dynamics of locomotion.

Collecting the data on real hardware instead is no escape. Scaling real-world data is hard enough for a fixed-base arm on a tabletop, and a humanoid raises both the price per trajectory and the number of trajectories required for sufficient coverage, as each demands extra supervision to ensure safe recovery and full-body balance restoration after failures. Simulation avoids these costs and further allows exploration of intermediate learning states that would be unsafe or impractical on hardware.

Recent work has begun to chip away at these limitations. MolmoSpaces [4] provides large-scale procedurally generated environments, reducing the diversity bottleneck, and MolmoBot [5] shows that policies trained on synthetic data from such environments transfer zero-shot to real-world tabletop manipulation under broad domain randomization. Other work trains humanoid systems purely in simulation [6, 7], showing that vision-based whole-body behaviors can also be deployed zero-shot to real hardware. These humanoid systems, however, are trained and evaluated within a heavily constrained visual environment structure, and rely on complex hand-crafted rewards. A vision-based humanoid loco-manipulation policy that generalizes across scene layouts without per-environment tuning remains an open problem.

Despite these advances, behavior cloning on synthetic data in simulation can be limited by two additional factors. The first is demonstration quality. Many demonstrations are inherently imperfect, and scripted simulated data in particular is prone to the imitation gap [8, 9], in which the demonstrator’s access to privileged information results in imperfect imitation. For loco-manipulation tasks in particular, this is caused by hidden phase variables producing discontinuous motion near the boundaries between navigation, reaching, and manipulation. And while such discontinuities could in principle be reduced, doing so demands per-task tuning that reintroduces the hand-engineering effort simulation is meant to spare us. The second factor is learning fidelity. Even given perfect demonstrations from a fully observable expert, behavior cloning cannot match that expert, because covariate shift compounds small errors into out-of-distribution states and because the imitation objective is not the task success objective.

In this work, we (1) study what scalable recipe is needed to train visual humanoid loco-manipulation policies in simulation, focusing on the factorization of locomotion and manipulation through a fixed low-level whole-body controller; (2) identify that scripted demonstration generation imposes a discrete phase structure whose boundaries are not observable from the policy’s inputs and are therefore hostile to imitation, and close this gap with group-relative policy optimization (GRPO) [10], which improves the policy beyond the demonstrator and is enabled by simulation’s identical-state group rollouts; (3) release **FetchMan-Bench**, a reproducible simulation benchmark for humanoid loco-manipulation; and (4) scale the resulting recipe to train **FETCHMAN**, an environment-generalist visual loco-manipulation policy learned entirely in simulation and deployed zero-shot on real hardware, further extending it to multi-object training as an initial step toward a task generalist.

2 Related Work

2.1 Synthetic Demonstration Generation

To overcome the data scarcity that robotics faces relative to the internet-scale corpora behind language models, synthetic data has been used across a wide variety of robotics tasks, including driving [11] and navigation [12, 13]. Generating manipulation data, however, is more demanding, since contact-rich behaviors

require high-precision physics that is harder to simulate faithfully. Harder still is pairing that fidelity with the scene diversity that visual generalization needs.

Early work [14] showed this transfer is possible, but with limited asset and scene diversity. InternVLA [15], GraspVLA [16], and MolmoBot [5] later scaled up the assets, transferring zero-shot from simulation to real-world tabletop manipulation. Behavior1K [17] and RoboCasa365 [18] target mobile manipulation, but on a fixed set of environments rather than scaling scene diversity toward real-world transfer. We generate demonstrations at a similar physical fidelity while scaling both scene and object diversity for a humanoid embodiment, and through a mixture of imitation and reinforcement learning.

2.2 Learning Humanoid Locomotion and Manipulation

A number of works train humanoid policies directly from teleoperation or human motion data. Human-Plus [19] retargets human motion to a humanoid embodiment and combines imitation learning (IL) with reinforcement learning (RL) fine-tuning. OmniH2O [20] enables universal whole-body teleoperation and transfers the resulting policies to real hardware. GR00T-N1 [21], Humanoid-VLA [22], and ψ_0 [23] train generalist VLA policies on heterogeneous mixtures of robot demonstrations and human video. These approaches are bottlenecked by the cost and scale of real-world data collection, which limits the diversity of scenes, objects, and failure modes they can cover, and often requires post-training adaptation at the deployment scene.

Simulation, on the other hand, enables large-scale data generation without hardware constraints. Recent systems such as VIRAL [6] and DoorMan [7] demonstrate that vision-based behaviors, a walk-place-grasp-turn cycle and articulated-door opening respectively, can be deployed zero-shot to real hardware. However, these policies are trained on a narrow environment structure, which limits their generalization to diverse real-world settings. ASAP [24] trains whole-body skills in simulation and refines them with real-world RL to close the sim-to-real gap, but targets a fixed set of motion skills rather than scene-level generalization.

2.3 Post-Training Flow-Based Policies

Policies initialized by imitation are commonly refined with RL to surpass the demonstrator. With the increased adoption of flow-based action heads, this has become harder, as such policies have no tractable action likelihood. Some methods leave the flow untouched and instead learn a policy over its starting noise [25]. Others like ReinFlow [26] and Flow-GRPO [27] add Gaussian noise to the sampler, turning the deterministic ODE into a stochastic process with closed-form per-step likelihoods. These methods have largely been applied to tabletop or simulated benchmark tasks. We adopt Flow-GRPO for humanoid loco-manipulation, refining entirely in simulation across a diverse distribution of scenes to close the imitation gap left by scripted demonstrations, without any real-world interaction.

3 Problem Setup

We use the Unitree G1 humanoid with Dex1-1 grippers, exposing a single control interface that accepts 15-dimensional whole-body commands at 10 Hz. Only the source of that command changes, from the scripted controller π_{ctrl} (Section 4.2) during data generation to the learned policy π_θ (Section 5.2) at deployment.

Observations. Our environment observation is $o_t = (I_t^{\text{head}}, I_t^{\text{wrist}}, o_t^{\text{prop}})$, where $I_t^{\text{head}}, I_t^{\text{wrist}} \in \mathbb{R}^{224 \times 384 \times 3}$ are a head-mounted fisheye RGB image and a wrist-mounted RGB image (Fig. 2), and the proprioceptive vector $o_t^{\text{prop}} = (h_t, r_t, p_t, q_t^{\text{upper}}, g_t)$ contains the base height h_t , base roll r_t and pitch p_t , upper-body joint positions q_t^{upper} (waist and right arm), and the gripper aperture g_t .

Actions. The 15-dimensional action is $a_t = (v_t, \bar{h}_t, \bar{q}_t^{\text{waist}}, \bar{q}_t^{\text{arm}}, \bar{g}_t)$, where $v_t = (v_x, v_y, \omega) \in \mathbb{R}^3$ is a base linear and yaw velocity command, $\bar{h}_t \in \mathbb{R}$ is the base-height target, $\bar{q}_t^{\text{waist}} \in \mathbb{R}^3$ and $\bar{q}_t^{\text{arm}} \in \mathbb{R}^7$ are waist and right-arm joint targets, and $\bar{g}_t \in \mathbb{R}$ is the right-gripper target.



Figure 2: **Simulated observations.** Example frames of the head-fisheye and wrist streams the policy receives, across scenes randomized in texture, lighting, and camera intrinsics and extrinsics.

Hierarchical Control. We adopt the action factorization standard in other humanoid loco-manipulation systems [6, 7, 23], where commands specify base velocity, base height, upper-body joint targets, and gripper aperture. The base command $(v_t, \bar{h}_t)$ is tracked by a pre-trained SONIC lower-body controller [28] running at 50 Hz, while the upper-body and gripper targets $(\bar{q}_t^{\text{waist}}, \bar{q}_t^{\text{arm}}, \bar{g}_t)$ are PD-tracked directly. Both resolve to torques at the 200 Hz physics rate. SONIC itself is a decoupled balance-and-walking pair, dispatched per step by a hard threshold on the commanded base velocity. This shrinks the command space from 29 joint torques to 15 meaningful commands, which is what makes both behavior cloning and RL fine-tuning feasible at our scale. And because the same stack runs in simulation and on hardware, its closed-loop response is a fixed property of the system rather than another source of sim-to-real shift.

4 FetchMan — Data Generation

We generate training data by rolling out a scripted, privileged whole-body controller in procedurally generated indoor scenes. Each episode samples a house, a pickable target object within it, a contact-free humanoid start pose, and a feasible grasp, after which the controller solves the task from privileged simulator state. The resulting trajectories pair the observations of Section 3 with 15-dimensional whole-body commands, and are directly usable for imitation (Fig. 3). Below we describe how tasks are sampled (Section 4.1), the scripted demonstrator (Section 4.2), the mixture of episode types we collect (Section 4.3), and the per-episode domain randomization that diversifies the data (Section 4.4).

4.1 Task Sampling

Episodes are instantiated using MolmoSpaces [4], an open ecosystem of procedurally generated indoor scenes populated with physically simulated objects. Our task sampler (Fig. 3) closely follows that of MolmoBot [5], which we extend to the humanoid setting. A task is a triple (s, m, x_0) of a scene, a target object, and a robot start pose.

Scene. A scene s is drawn from a pool spanning the ProcTHOR-10k, Holodeck, and ProcTHOR-Objaverse splits curated by MolmoSpaces. Once loaded, we precompute a 2D free-space occupancy grid $\mathcal{O}_s$ that dilates the static collision geometry by the G1’s footprint radius. $\mathcal{O}_s$ is used both to reject infeasible base placements and to plan navigation waypoints in Section 4.2.

Target object. The target m is drawn from the free-jointed assets marked by MolmoSpaces as *pickable*. Each instance carries a grasp set $\mathcal{G}_m$, precomputed in MolmoSpaces by antipodal sampling and post-filtering and expressed in the object frame, which we further filter by our Dex1-1 gripper’s geometry and by G1 IK feasibility. The object’s pose T_m and the height of its supporting surface are resampled every episode (Section 4.4).

Robot start pose. Candidate base positions are sampled from the free cells of $\mathcal{O}_s$ in an annulus $r_{\min} \leq \|p - p_m\|_2 \leq r_{\max}$ around the target, with yaw set to the bearing-to-target plus an offset $\Delta\psi \sim \mathcal{U}(-0.5, 0.5)$ rad. We accept the first candidate x_0 whose whole-body pose is contact-free and from which the target is visible

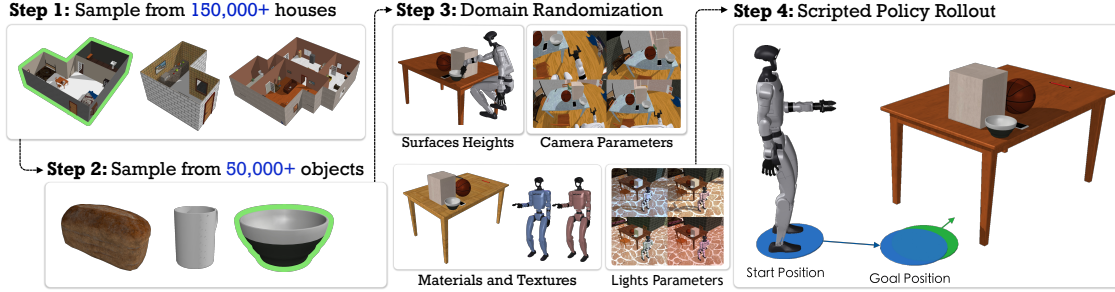


Figure 3: **Synthetic data generation pipeline.** Each episode samples (1) a house from MolmoSpaces, (2) a pickable target object within it, (3) a contact-free start and grasp-standoff goal pose connected by a planned path the scripted controller follows, and (4) domain randomization applied independently per episode.

in the head fisheye camera. Crucially, we validate x_0 against the full simulator rather than against $\mathcal{O}_s$. A flat footprint conservatively rejects any base pose whose 2D area overlaps a support surface, but the G1’s upper body can freely occupy the volume above that surface, e.g., the hand reaching over a table. Validating in 3D lets $r_{\min}$ shrink well below what static tabletop setups assume is safe, widening the distribution of approach geometries and better matching the clutter of real-world deployment.

4.2 Scripted Demonstrator

Each accepted task is rolled out by a scripted whole-body controller π_{ctrl} , which acts on privileged simulator state including the object pose T_m , a selected grasp T_g , the occupancy grid $\mathcal{O}_s$, the robot’s base pose x_t , and its own phase index k_t . None of these is visible to π_θ (Section 5.2), which sees only o_t . The demonstrations therefore share the action interface of Section 3 but not its information interface.

Reaching. While the robot is outside the grasp standoff, π_{ctrl} plans an A* path on an extra-inflated copy of $\mathcal{O}_s$, reduces it to a short list of straight-line waypoints, and emits base velocity commands (v_x, v_y, ω) that drive the robot toward the next waypoint while keeping the chassis facing the target.

Manipulation. On arrival at the grasp standoff, π_{ctrl} advances through a discrete phase sequence $k_t \in \{\text{reach, descend, close, lift}\}$, moving to the next once the end-effector reaches the current phase’s target. The grasp T_g is selected once per episode. We transform each candidate in $\mathcal{G}_m$ into world coordinates through T_m and take a random grasp that is reachable under Mink [29] inverse kinematics with self-collision avoidance enabled. The IK treats the base height $\bar{h}$ as a free variable jointly optimized with the arm joints, so π_{ctrl} can drop or raise the torso to bring T_g into the arm’s dexterous workspace rather than committing to a fixed standing height. Finally, we discard episodes with invalid paths, infeasible grasps, or phase timeouts.

4.3 Episode Mixture

We collect two episode types under a fixed mixture. 80% are *fetch* (reach+pick) episodes that exercise the full navigation-to-grasp pipeline, and 20% are *pick-only* episodes in which the robot is initialized directly at a grasp standoff and only the manipulation phases are executed. Beyond re-balancing the data toward the terminal grasp, the pick-only episodes let us randomize the initial upper-body configuration, which the fetch episodes cannot. We sample the arm joints around their defaults and draw the initial gripper aperture uniformly between fully open and fully closed. A fetch episode, by contrast, must arrive at the standoff from the default arms-forward, gripper-open state, so a policy trained on those alone would meet any deviation from it out of distribution.

4.4 Domain Randomization

On top of the task instance, every episode independently resamples the following axes (Fig. 3).

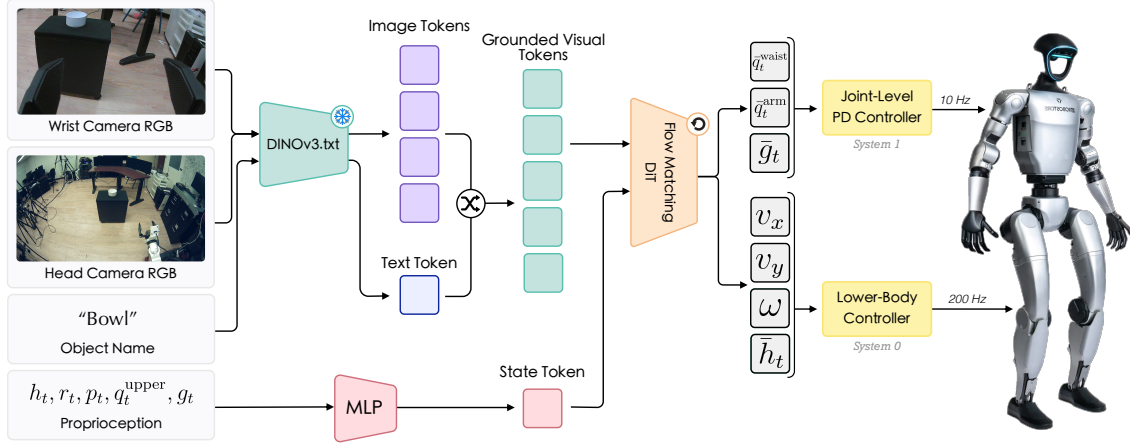


Figure 4: **Policy architecture.** The head and wrist RGB streams are encoded by a frozen DINOv3 backbone into per-patch image tokens, while the proprioceptive vector $(h_t, r_t, p_t, q_t^{\text{upper}}, g_t)$ is mapped by an MLP to a single state token. A flow-matching DiT cross-attends to these tokens and denoises a 15-dimensional action chunk, split into upper-body targets (waist, arm, and gripper) tracked by a joint-level PD controller and a base command $(v_x, v_y, \omega, \bar{h}_t)$ executed by the pre-trained lower-body locomotion controller.

- **Textures.** Scene surfaces (walls, floors, counters, tables, and doors) draw textures from predefined material pools.
- **Lighting.** Each light source’s position, color, intensity, and shadow toggle are resampled, and the robot’s body color is tinted around its stock white.
- **Cameras.** The head fisheye and wrist cameras are perturbed in both intrinsics (field of view, fisheye distortion coefficients) and extrinsics (position and orientation offsets).
- **Surface and object pose.** The target’s supporting surface is reset to a height within the G1’s reachable workspace and propagated rigidly to every object resting on it via a contact-graph traversal. The target’s planar pose on that surface is then jittered by a small Gaussian.
- **Action noise.** Small zero-mean Gaussian noise is added to each command before it reaches the low-level controllers, so the policy cannot rely on exact open-loop replay of the demonstrator.

5 FetchMan — Policy Learning

We train our unprivileged policy $\pi_\theta(a|o)$ in two stages (Fig. 5). In **Stage 1** (Section 5.2), we behavior-clone the scripted controller π_{ctrl} of Section 4.2, giving π_θ a usable prior for a high-dimensional, partially observable control problem. In **Stage 2** (Section 5.3), we fine-tune π_θ with large-scale on-policy RL under a sparse task-success reward. This repairs the structural artifact that π_{ctrl} ’s discrete phase machine leaves on π_θ during behavior cloning, which forces the policy to learn an implicit phase classifier from images and proprioception in lieu of the hidden state π_{ctrl} uses internally to advance its phase. The resulting checkpoint is deployed zero-shot on the real robot.

5.1 Architecture

Fig. 4 summarizes our model’s architecture. The vision encoder is a frozen DINOv3 ViT-B/16 [30] that emits a patch-token grid per camera. The proprioceptive vector o_t^{prop} is projected into a single state token that is appended to the visual context. The action head is a DiT-style transformer [31] that attends from noised action tokens to the observation context via cross-attention, and predicts a chunk of actions $a_{t:t+H}$.

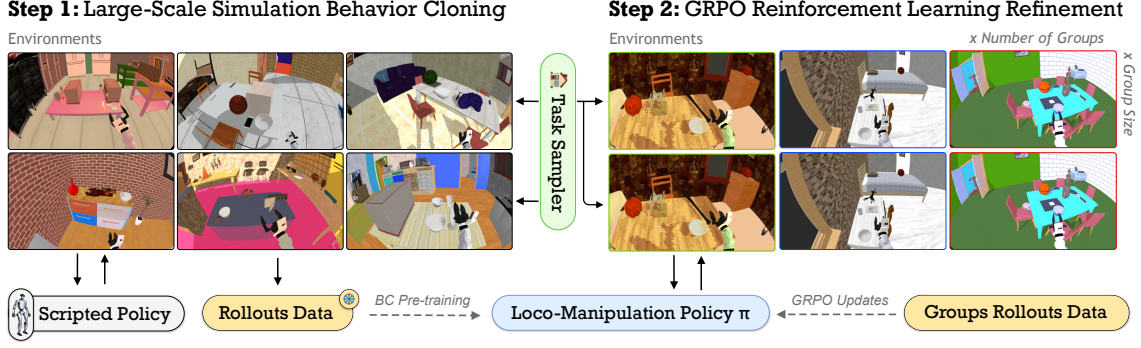


Figure 5: **Two-stage training pipeline.** *Stage 1 (Imitation Pretraining).* The task sampler instantiates diverse simulated environments in which the scripted G1 staged policy generates demonstration trajectories, and the resulting frozen dataset trains the loco-manipulation policy π by behavior cloning. *Stage 2 (Online GRPO Refinement).* The same sampler resets groups of environments to identical tasks (group size $\times$ number of groups), π collects on-policy trajectories that differ only in sampled action noise, and a group-relative GRPO update refines π in place.

Text conditioning. For the multi-object variant of the model we train, we swap the backbone for a frozen dino.txt [32] stack, whose vision and text towers are pre-aligned to a shared space. The object name is encoded into a text token that cross-attention fuses into the patch tokens, yielding *grounded* visual tokens that replace them in the context.

5.2 Imitation Pretraining

We pretrain π_θ by behavior cloning on the trajectories of Section 4. Each training sample is a window consisting of the observation o_t and the demonstrator’s next H action commands $a_{t:t+H}$. Following π_0 [33], we re-parameterize the 11 dimensions corresponding to absolute joint or pose targets (base height, waist, right arm) as deltas relative to a per-chunk anchor read from the observation at t , leaving the base velocity command and the gripper command absolute. We use $H = 16$, and at deployment only the first 8 actions of each chunk are executed before a new prediction is generated from the latest observation.

5.3 Flow-GRPO Refinement

Dagger [34] does not apply cleanly here, as the scripted expert acts on a phase variable it reads from privileged state, so relabeling the policy’s visited states still yields actions conditioned on state the policy never observes. We therefore refine π_θ from the imitation checkpoint with Flow-GRPO [27], the extension of group-relative policy optimization [10] to flow-matching policies. At inference we replace the deterministic Euler step of the flow integrator with a Gaussian transition $x_{\tau+\Delta\tau} \sim \mathcal{N}(x_\tau + \hat{v}_\theta(x_\tau, \tau, o_t) \Delta\tau, \sigma_\tau^2 |\Delta\tau|)$ whose noise schedule $\sigma_\tau = \eta\sqrt{(1-\tau)/\tau}$ matches the rectified-flow schedule, so the marginal at $\tau=1$ remains the policy’s data distribution and the per-step log-likelihood is a closed-form Gaussian density. The same schedule is used for rollouts and for gradient passes, so the per-step importance ratio between the current and rollout-time policies is well defined.

Each update collects G groups of M episodes. Every group is reset to a single task sample and rolled out M times from that same state, so episodes within a group differ only in the SDE noise injected by π_θ . We use $M=8$ throughout, with $G=32$ groups (256 episodes per update) for the single-object policy and $G=64$ (512 episodes per update) for the multi-object policy of Section 6.6. For a group with returns $\{R_i\}$, the advantage is the standardized group-relative return $A_i = (R_i - \mu)/\sigma$, and no learned value function is used. Groups that are entirely successful or entirely unsuccessful carry no signal and are dropped, so only mixed groups contribute gradient. The actor is updated with a PPO-style clipped objective [35] over the per-flow-step importance ratio, with dual-clipping to bound the loss from below on large negative-advantage samples, and

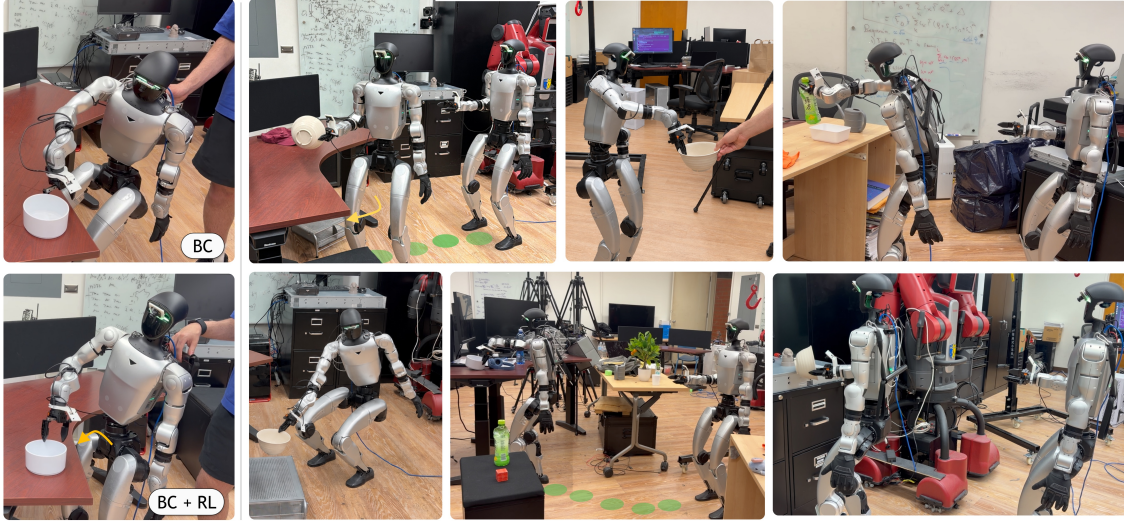


Figure 6: **Real-world rollouts and the effect of RL refinement.** *Left.* Representative zero-shot rollouts of FetchMan on the Unitree G1 across diverse, unseen scenes, where the policy navigates to the target bowl and grasps it from head- and wrist-camera observations alone. *Right.* The locomotion-to-manipulation handoff before and after Flow-GRPO refinement, on a fixed initialization. The behavior-cloned policy grasps as soon as the target is nearby and misses from too far, while after RL it walks in until the object is within reach and only then grasps.

optionally a KL penalty to the frozen behavior-cloned policy ($\beta=0$ for the single-object policy, $\beta=0.01$ for the multi-object policy). Identical-state group rollouts are essentially free in simulation and infeasible in the real world, which is what makes this within-group baseline viable even at fairly small group sizes.

The return R_i is a single sparse signal, 1 for a successful fetch of the target and 0 otherwise, keeping the objective aligned with the deployment metric without per-step shaping to tune across our scene and object distribution. Since successes terminate at the lift while failures run to the time limit, we length-normalize the actor loss by $\bar{L}/L_i$ (episode length L_i , batch mean $\bar{L}$) so each episode, rather than each transition, contributes equally and long failures do not dominate the update.

6 Experiments

Our experiments are organized around four questions.

- RQ1.** Is scaling synthetic loco-manipulation demonstrations sufficient, or does behavior cloning saturate below the demonstrator?
- RQ2.** Does RL refinement recover the headroom that scaling synthetic data leaves behind?
- RQ3.** Does the resulting policy transfer zero-shot to a real Unitree G1, and which design choices are necessary for it to do so?
- RQ4.** Can the recipe extend beyond a single object toward object-generalist loco-manipulation?

6.1 Setup

Instantiating the data-generation pipeline of Section 4, we collect 150k bowl-pick demonstrations, roughly 650 hours of robot experience across $\sim 150k$ distinct procedurally generated scenes, at ~ 100 episodes per minute on a single NVIDIA L40S (~ 40 GPU-hours in total). Unless stated otherwise, our reference

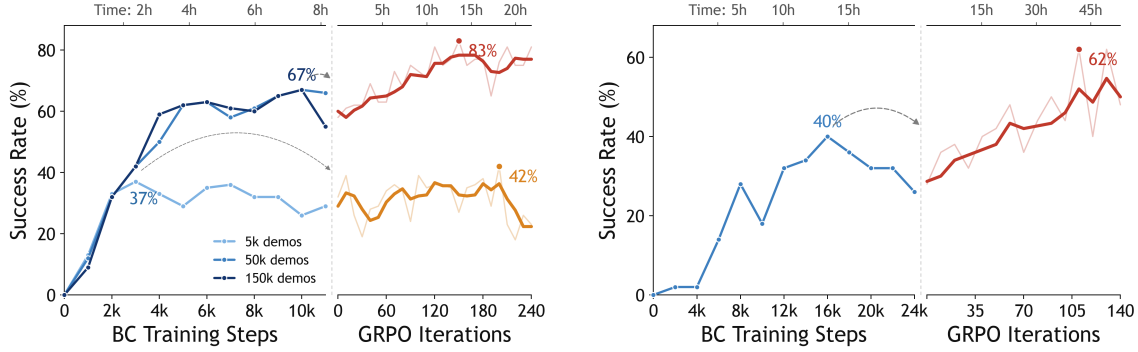


Figure 7: **Behavior cloning saturates, RL breaks through.** *Left.* Single-object loco-manipulation SR (%) on the 100-scene validation set. The first segment scales BC demonstrations (5k–150k) and plateaus after 50k, while the second continues the 150k checkpoint with GRPO, which pushes past the imitation ceiling. A second run applies the same refinement to the 5k checkpoint and does not improve comparably. *Right.* The text-conditioned multi-object policy, whose BC success rate peaks near 40% before GRPO lifts it to roughly 62%. Time in GPU hours.

model uses the frozen DINOv3 encoder, delta-action targets, and the full 150k-demonstration set. We call it FetchMan-BC before RL refinement and FetchMan (BC+RL) after.

6.2 FetchMan-Bench

We evaluate on FETCHMAN-BENCH, our reproducible simulation benchmark for humanoid loco-manipulation, with fixed held-out scenes and scoring released so results are comparable across methods. It reports two success rates. *Manipulation* SR isolates the terminal phase, where the robot is initialized at a grasp standoff and must grasp and lift the target. *Loco-manipulation* SR measures the full task, in which the robot starts away from the object and must navigate to it before grasping, additionally stressing the locomotion-to-manipulation handoff at which π_{ctrl} crosses its hidden phase boundary. In simulation we evaluate on a fixed set of 100 initializations never seen during training and report the best validation checkpoint, which is then deployed zero-shot on the physical G1. No real-world data is used at any stage. Standard errors assume a binomial distribution.

6.3 Scaling Alone Saturates Below the Demonstrator (RQ1)

We first ask whether more demonstrations are enough. The left segment of Fig. 7 scales the behavior-cloning set from 5k to 150k demonstrations. Loco-manipulation SR climbs steeply from 5k to 50k (40% to 67%) and then *saturates*, as the further jump to 150k yields no measurable gain (67%). This plateau is the empirical signature of the imitation ceiling argued in Section 5. Because π_{ctrl} ’s phase boundaries are driven by a hidden index k_t that the policy cannot observe, additional demonstrations from the same demonstrator cannot teach the policy what it structurally cannot see. Scaling data alone is therefore insufficient.

In principle this ceiling could be raised by engineering a demonstrator with smoother, more varied phase transitions, but doing so reintroduces the per-task, hand-tuned controller design that scaling is meant to avoid. Our RL stage instead requires only a single sparse success reward. We therefore treat the ceiling as intrinsic to any fixed scripted demonstrator and move past it with reinforcement learning (Section 6.4).

6.4 RL Refinement Recovers the Headroom (RQ2)

The right segment of Fig. 7 continues the saturated 150k checkpoint with Flow-GRPO. RL breaks through the plateau, lifting simulated loco-manipulation SR from 67% to 83%, confirming that the remaining

Method	Simulation SR [%]		Real-World SR [%]	
	Manipulation	Loco-Manipulation	Manipulation	Loco-Manipulation
Num. Trials	100	100	22	30
FetchMan-BC	75.0 $\pm$ 4.3	67.0 $\pm$ 4.7	72.7 $\pm$ 9.5	56.7 $\pm$ 9.0
FetchMan (BC+RL)	79.0 $\pm$ 4.1	83.0 $\pm$ 3.8	77.2 $\pm$ 8.9	73.3 $\pm$ 8.1

Table 1: **Main results.** RL refinement improves the full loco-manipulation task in both simulation and the real world while leaving the already-saturated terminal grasp essentially unchanged. Values are success rates (SR) in [%] $\pm$ standard error.

Variant	Simulation SR [%]		Real-World SR [%]	
	Manipulation	Loco-Manipulation	Manipulation	Loco-Manipulation
FetchMan-BC (DINOv3, delta)	75.0 $\pm$ 4.3	67.0 $\pm$ 4.7	72.7 $\pm$ 9.5	56.7 $\pm$ 9.0
with SigLIP	58.0 $\pm$ 4.9	42.0 $\pm$ 4.9	8.3 $\pm$ 8.0	0.0
with absolute actions	62.0 $\pm$ 4.9	45.0 $\pm$ 5.0	16.7 $\pm$ 10.8	0.0

Table 2: **Architecture ablations.** A frozen DINOv3 encoder and delta-action targets are each necessary for zero-shot transfer; replacing either collapses real-world performance. Values are success rates (SR) in [%] $\pm$ standard error, over the reference BC configuration. Simulation is over 100 held-out initializations; real-world trials are 22 manipulation and 30 loco-manipulation for the reference, and 12/10 respectively for the ablation variants.

headroom comes from a different learning signal rather than from more data, one that lets the policy relax the demonstrator’s discrete phase boundaries rather than imitate them.

The same refinement is far less effective from a weaker initialization. The second trajectory in Fig. 7 applies identical Flow-GRPO settings to the 5k-demonstration checkpoint (40%), and over the same number of iterations it oscillates around its starting success rate, peaking at 42% without a clear upward trend. RL refinement therefore does not substitute the scale of imitation pretraining needed. The group-relative advantage needs a prior that already succeeds often enough for groups to be mixed, and a policy near the floor yields mostly all-failure groups that are dropped by the filter.

Table 1 isolates where these gains land. RL raises loco-manipulation SR in both simulation (67% to 83%) and the real world (56.7% to 73.3%), while manipulation SR is nearly unchanged (72.7% to 77.2% in the real world). The terminal grasp is already well cloned, and RL’s gains concentrate almost entirely in the loco-manipulation regime, exactly the phase-transition behavior that Section 5 predicts behavior cloning cannot recover from privileged-state-driven demonstrations.

6.5 Zero-Shot Transfer and What Enables It (RQ3)

The reference checkpoint is deployed zero-shot on the physical G1, with no real-world data at any stage. Fig. 6 shows representative rollouts across diverse, unseen scenes, where the policy navigates to the target and grasps it from head- and wrist-camera observations alone. The final FetchMan (BC+RL) model reaches 73.3% loco-manipulation SR on hardware (Table 1), establishing that the simulation-trained recipe transfers.

Table 2 identifies what makes this transfer possible, holding everything else at the reference configuration. Two choices dominate. The first is the vision encoder. A frozen DINOv3 encoder substantially outperforms SigLIP in simulation (67% vs. 42%), and in the real world SigLIP collapses almost entirely (8.3% manipulation, 0% loco-manipulation). We attribute this to DINOv3’s stronger geometric features, which cross the sim-to-real appearance gap without finetuning. The second is the action parameterization. Absolute targets, trained for the same number of steps, reach only 16.7% manipulation and 0% loco-manipulation on hardware, confirming that the near-zero-mean delta targets helps in the transfer.

6.6 Toward Object Generalism (RQ4)

Finally, we ask whether the same recipe reaches beyond a single object. We regenerate data across all pickable object categories, collecting 350k demonstrations under an identical pipeline, and train the text-conditioned variant of Section 5.1 that grounds a dino.txt backbone on the target object’s name. We behavior-clone and then refine this model with Flow-GRPO, as shown in Fig. 7, and Fig. 8 shows example rollouts across object categories. Because this policy’s behavior-cloned prior succeeds less often than the single-object one, a larger fraction of groups return entirely unsuccessful and are discarded by the mixed-group filter of Section 5.3, so we double the group count to $G=64$ to retain a comparable number of gradient-carrying groups per update. In simulation, the multi-object BC policy reaches 40% and the fine-tuned policy 62%

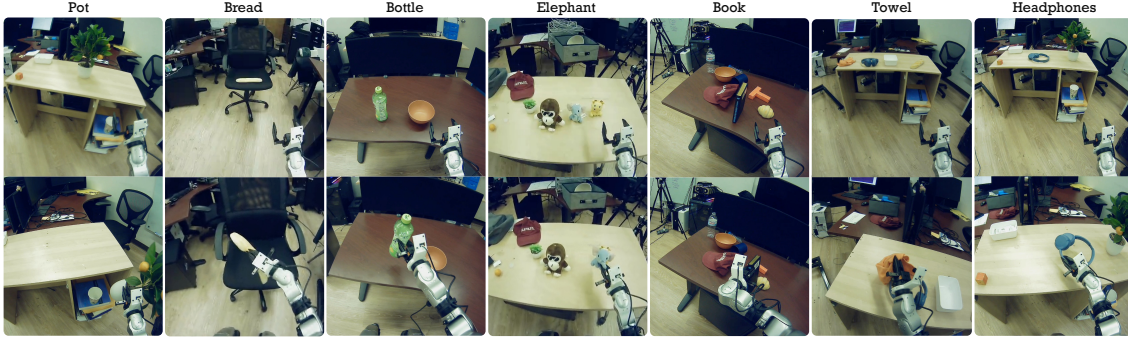


Figure 8: **Multi-object rollouts.** Head-camera views from the text-conditioned policy grasping different target objects, one episode per row, with the prompted object name shown at left.

loco-manipulation SR, below the single-object performance of 83%. Some of this difference is expected given the more difficult task distribution.¹

The lower performance of the base multi-object policy relative to the BC-only MolmoBot [5] policy suggests that substantial gains could come from switching from an LBM-style [36] architecture to a VLA-style one. The dino.txt backbone is frozen and comparatively small, and a larger pre-trained vision-language model used as the encoder would bring stronger grounding of the target name to its visual referent. Finetuning the backbone jointly with the policy at a low learning rate is a further option. We did not pursue these strategies, as they are substantially more computationally expensive and our compute and hardware access were limited.

We did, however, run qualitative zero-shot evaluations on the real G1. The policy produces successful fetches across several object categories, though it is noticeably less robust than the single-object policy. This answers our research question positively, in that the recipe can extend to the multi-object setting, with further scaling left to future work. Videos are shared on our website.

7 Limitations

Stateless policy. Our π_θ acts on a single observation o_t with no history, so the demonstrator’s phase index k_t must be inferred from one frame. A short observation history would likely make the phase partially recoverable, but stacking frames multiplies the visual token count and the cost of every training step, and we did not have the compute budget to ablate it here.

Fixed lower-body controller. π_θ commands SONIC through base velocity and height and never adapts it. This decoupling makes learning tractable at our scale but fixes the robot’s gait and balance behavior. The policy cannot brace against heavy objects, widen its stance to extend reach, or recover from disturbances outside SONIC’s envelope. Unfreezing the lower body would return the policy to a 29-dimensional torque space and require relearning balance alongside manipulation, at a scale cost beyond our budget.

Task diversity. Our tasks are limited to fetch (reach-and-pick). MolmoSpaces supports articulated-object manipulation, and longer-horizon behaviors could be composed from atomic skills of this kind. We leave this to future work.

8 Conclusion

We presented FetchMan, which learns visual humanoid loco-manipulation entirely in simulation and deploys it zero-shot to a real Unitree G1. A scripted controller generates demonstrations across hundreds of thousands

¹On its own data, which used a similar object distribution, MolmoBot reports 100% success on bowls against 66.8% on diverse objects.

of scenes and objects, a flow-matching policy is behavior-cloned on them, and group-relative RL refines it beyond the demonstrator. Behavior cloning saturates well below the achievable ceiling, which we attribute to the demonstrator’s unobservable phase structure, and RL recovers the remaining headroom in both simulation and the real world. A preliminary text-conditioned model extends the recipe to multiple object categories, taking a first step toward object generalism.

Acknowledgments

We thank Nirvana for providing the Unitree G1 robot used in our experiments, and Mahi Shafiullah for helpful discussions throughout.

References

- [1] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.
- [2] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel. mjlal: A lightweight framework for gpu-accelerated robot learning. *arXiv preprint arXiv:2601.22074*, 2026.
- [3] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Munoz, X. Yao, R. Zurbrugg, N. Rudin, et al. Isaac lab: A gpu-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025.
- [4] Y. Kim, W. Pumacay, O. Rayyan, M. Argus, W. Han, E. VanderBilt, J. Salvador, A. Deshpande, R. Hendrix, S. Jauhri, et al. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*, 2026.
- [5] A. Deshpande, M. Guru, R. Hendrix, S. Jauhri, A. Eftekhari, R. Tripathi, M. Argus, J. Salvador, H. Fang, M. Wallingford, et al. Molmob0t: Large-scale simulation enables zero-shot manipulation. *arXiv preprint arXiv:2603.16861*, 2026.
- [6] T. He, Z. Wang, H. Xue, Q. Ben, Z. Luo, W. Xiao, Y. Yuan, X. Da, F. Castañeda, S. Sastry, C. Liu, G. Shi, L. Fan, and Y. Zhu. Viral: Visual sim-to-real at scale for humanoid loco-manipulation. *arXiv preprint arXiv:2511.15200*, 2025.
- [7] H. Xue, T. He, Z. Wang, Q. Ben, W. Xiao, Z. Luo, X. Da, F. Castañeda, G. Shi, S. Sastry, L. J. Fan, and Y. Zhu. Opening the sim-to-real door for humanoid pixel-to-action policy transfer. *arXiv preprint arXiv:2512.01061*, 2025. URL <https://arxiv.org/abs/2512.01061>.
- [8] L. Weihs, U. Jain, I.-J. Liu, J. Salvador, S. Lazebnik, A. Kembhavi, and A. Schwing. Bridging the imitation gap by adaptive insubordination. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19134–19146. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/9fc664916bce863561527f06a96f5ff3-Paper.pdf.
- [9] L. Nguyen, M. Fauth, B. Jaeger, D. Dauner, M. Igl, A. Geiger, and K. Chitta. Lead: Minimizing learner-expert asymmetry in end-to-end driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
- [10] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. Li, Y. Wu, et al. Deepseek-math: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

- [11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In S. Levine, V. Vanhoucke, and K. Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 1–16. PMLR, 13–15 Nov 2017. URL <https://proceedings.mlr.press/v78/dosovitskiy17a.html>.
- [12] K. Ehsani, T. Gupta, R. Hendrix, J. Salvador, L. Weihs, K.-H. Zeng, K. P. Singh, Y. Kim, W. Han, A. Herrasti, R. Krishna, D. Schwenk, E. VanderBilt, and A. Kembhavi. Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. *arXiv*, 2023.
- [13] A. Eftekhari, R. Hendrix, L. Weihs, J. Duan, E. Caglar, J. Salvador, A. Herrasti, W. Han, E. VanderBil, A. Kembhavi, et al. The one ring: a robotic indoor navigation generalist. *arXiv preprint arXiv:2412.14401*, 2024.
- [14] S. James, A. J. Davison, and E. Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. In *Conference on Robot Learning*, pages 334–343. PMLR, 2017.
- [15] J. Cai, Z. Cai, J. Cao, Y. Chen, Z. He, L. Jiang, H. Li, H. Li, Y. Li, Y. Liu, et al. Internvla-a1: Unifying understanding, generation and action for robotic manipulation. *arXiv preprint arXiv:2601.02456*, 2026.
- [16] S. Deng, M. Yan, S. Wei, H. Ma, Y. Yang, J. Chen, Z. Zhang, T. Yang, X. Zhang, H. Cui, Z. Zhang, and H. Wang. Graspvla: a grasping foundation model pre-trained on billion-scale synthetic action data. In J. Lim, S. Song, and H.-W. Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 1004–1029. PMLR, 27–30 Sep 2025. URL <https://proceedings.mlr.press/v305/deng25a.html>.
- [17] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun, M. Anvari, M. Hwang, M. Sharma, A. Aydin, D. Bansal, S. Hunter, K.-Y. Kim, A. Lou, C. R. Matthews, I. Villa-Renteria, J. H. Tang, C. Tang, F. Xia, S. Savarese, H. Gweon, K. Liu, J. Wu, and L. Fei-Fei. Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 80–93. PMLR, 14–18 Dec 2023. URL <https://proceedings.mlr.press/v205/li23a.html>.
- [18] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations (ICLR)*, 2026.
- [19] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn. Humanplus: Humanoid shadowing and imitation from humans. *arXiv preprint arXiv:2406.10454*, 2024.
- [20] T. He, Z. Luo, X. He, W. Xiao, C. Zhang, W. Zhang, K. Kitani, C. Liu, and G. Shi. Omnih2o: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. *arXiv preprint arXiv:2406.08858*, 2024.
- [21] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, L. Fan, Y. Fang, D. Fox, F. Hu, S. Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.
- [22] P. Ding, J. Ma, X. Tong, B. Zou, X. Luo, Y. Fan, T. Wang, H. Lu, P. Mo, J. Liu, et al. Humanoid-vla: Towards universal humanoid control with visual integration. *arXiv preprint arXiv:2502.14795*, 2025.
- [23] S. Wei, H. Jing, B. Li, Z. Zhao, J. Mao, Z. Ni, S. He, J. Liu, X. Liu, K. Kang, et al. ψ_0 : An open foundation model towards universal humanoid loco-manipulation. *arXiv preprint arXiv:2603.12263*, 2026.

- [24] T. He, J. Gao, W. Xiao, Y. Zhang, Z. Wang, J. Wang, Z. Luo, G. He, N. Sobanbab, C. Pan, et al. Asap: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. *arXiv preprint arXiv:2502.01143*, 2025.
- [25] A. Wagenmaker, M. Nakamoto, Y. Zhang, S. Park, W. Yagoub, A. Nagabandi, A. Gupta, and S. Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.
- [26] T. Zhang, C. Yu, S. Su, and Y. Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38:106282–106319, 2026.
- [27] J. Liu, G. Liu, J. Liang, Y. Li, J. Liu, X. Wang, P. Wan, D. Zhang, and W. Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026.
- [28] Z. Luo, Y. Yuan, T. Wang, C. Li, F. Castañeda, S. Chen, Z.-A. Cao, J. Li, D. Minor, Q. Ben, J. Park, D. Sami, Z. Wang, X. Da, R. Ding, C. Hogg, L. Song, E. Lim, E. Jeong, T. He, H. Xue, W. Xiao, S. Yuen, J. Kautz, Y. Chang, U. Iqbal, L. Fan, and Y. Zhu. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *arXiv preprint arXiv:2511.07820*, 2025.
- [29] K. Zakka. Mink: Python inverse kinematics based on mujoco. URL <https://github.com/kevinzakka/mink>, 10:2, 2024.
- [30] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.
- [31] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.
- [32] C. Jose, T. Moutakanni, D. Kang, F. Baldassarre, T. Darcet, H. Xu, D. Li, M. Szafraniec, M. Ramamonjisoa, M. Oquab, et al. Dinov2 meets text: A unified framework for image-and pixel-level vision-language alignment. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24905–24916, 2025.
- [33] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. *pi_0*: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.
- [34] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.
- [35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [36] J. Barreiros, A. Beaulieu, A. Bhat, R. Cory, E. Cousineau, H. Dai, C.-H. Fang, K. Hashimoto, M. Z. Irshad, M. Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *Science Robotics*, 11(113):eaea6201, 2026.