Learning Visual Humanoid Loco-Manipulation Policies from Simulated Experiences

Author Names Omitted for Anonymous Review.

***Abstract*—Generalist visual humanoid policies that combine locomotion and manipulation across diverse settings, without per-environment tuning, represent a central aim of current robotics research. Achieving this requires demonstration data at a scale that real-world teleoperation still struggles to supply. Imitation learning from simulation (sim2real) is an emerging paradigm to address this, but it can suffer from a performance ceiling caused by two fundamental limitations: the imitation gap in which the demonstrator limits achievable quality, e.g., by scripted controllers such as state machines embedding a hidden phase structure that is not optimal, and learning fidelity issues leaving a further performance gap below even that ceiling due to imperfect learning and compounding error. We present ViLMa, an end-to-end sim2real pipeline that generates large-scale synthetic loco-manipulation demonstrations spanning tens of thousands of scenes and objects from a scripted whole-body controller, behavior-clones a policy on these demonstrations, and then closes the imitation gap via reinforcement learning using GRPO. We validate ViLMa by using it to train a humanoid reach and pick policy that walks to and grasps a target across diverse, unseen indoor scenes. Deployed zero-shot on a real Unitree G1, it reaches a 73% success rate in our tested scenes, and, to our knowledge, is the first visual humanoid loco-manipulation environment-structure generalist learned entirely in simulation.**

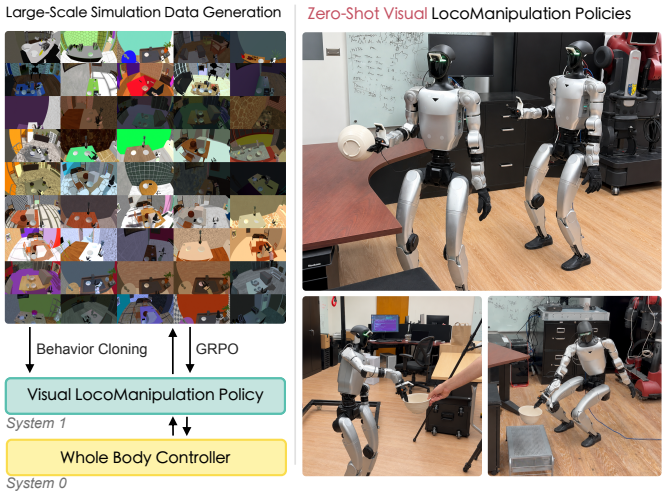

Fig. 1. ViLMa learns visual humanoid loco-manipulation entirely in simulation and deploys it zero-shot to a real Unitree G1. The resulting policy walks up to and grasps objects across diverse, unseen real-world scenes, with no real-world data or per-scene tuning.

I. Introduction

Simulation has become the standard setting for training humanoid locomotion policies. In these tasks, interaction with the environment is typically limited to feet-floor contact. Within minutes, a walking, dancing, or back-flipping policy can be brought to life using easy-to-use tools [1, 2, 3]. This level of success, however, has yet to translate to loco-manipulation due to (1) the limited availability of diverse simulated data, which requires a significant engineering effort to scale, and (2) the complexity of modeling contact-rich interactions, which is inherently more difficult than than the relatively self-contained dynamics of locomotion.

Collecting the scale of real-world data required for humanoid loco-manipulation is particularly costly. Each trajectory demands hardware supervision, safe recovery and reset after failures, full-body balance restoration, and sufficient variation across environments, support surfaces, object poses, lighting conditions, and rare interaction states. Simulation is attractive not only because it reduces collection costs but also because it allows exploration of intermediate learning states that would be unsafe or impractical to reach on physical hardware. The challenge, however, is generating synthetic data that simultaneously captures scene diversity, physically realistic contact interactions, and a control interface that transfers reliably to the real robot.

Recent work in manipulation 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 in such environments can transfer zero-shot to real-world tabletop manipulation. This work addresses a distinct problem: sim-to-real transfer gaps between the training and deployment distributions mean the policy encounters states at test time that differ systematically from simulation.

Humanoid systems trained in simulations, such as VIRAL [6] and DoorMan [7] demonstrate that vision-based behaviors can be deployed zero-shot to real hardware but consider only narrow environment distributions. A humanoid loco-manipulation policy that can handle multiple objects and scenes without environment-specific finetuning remains an unsolved challenge.

Despite these advances, behavior cloning performance can be limited by two additional factors. First, demonstration quality, Many demonstrations are inherently imperfect, and scripted simulated data in particular is prone to the imitation gap [8, 9] in which access to privileged information results in imperfect imitation. For manipulation tasks, this could easily be caused by hidden phase variables causing discontinuous motion near the boundaries between navigation, reaching, grasping, and lifting. More demonstrations from the same demonstrator reduce variance but cannot remove this

fundamental information mismatch.

A second factor is learning fidelity: even given perfect demonstrations from a fully observable demonstrator, behavior cloning cannot match the demonstrator due to covariate shift — compounding errors drive the policy into out-of-distribution states — and due to mismatches between the imitation objective and the task success objective.

In this work, we (1) study what recipe is needed to train visual humanoid loco-manipulation policies that are stable and scalable, 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; we close this gap with group-relative policy optimization (GRPO) [10] that improves the policy beyond the demonstrator, enabled by simulation's identical-state group rollouts; and (3) scale the resulting recipe into what, to our knowledge, is the first *environment-structure generalist* visual humanoid loco-manipulation policy learned entirely from simulated experience and deployed zero-shot to real hardware.

II. Related Work

A. 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, mainly driving [11], robot navigation [12]. Generating manipulation data however is more demanding, since contact-rich behaviors such as solving a Rubik's cube [13] require high-precision physics that is harder to simulate faithfully. The harder requirement still is pairing this fidelity with the scene diversity that generalization needs.

To overcome the data scarcity that robotics faces relative to the internet-scale corpora available to language models, synthetic data has been widely adopted in domains such as autonomous driving [11] and robot navigation [12]. Extending this to manipulation is substantially more challenging given the diversity and high-fidelity physic needed.

InternVLA [14], GraspVLA [15], and MolmoBot [5] pursue this combination, with the latter transferring zero-shot from simulation to real-world tabletop manipulation. Behavior1K [16] and RoboCasa365 [17] target mobile manipulation, but on a fixed set of environments rather than scaling scene diversity, and on non-humanoid platforms. We generate demonstrations at a similar physical fidelity while scaling both scene and object diversity for a humanoid embodiment.

B. Learning Humanoid Locomotion and Manipulation

A number of works train humanoid policies directly from teleoperation or human motion data. HumanPlus [18] retargets human motion to a humanoid embodiment and combines imitation learning (IL) with reinforcement learning (RL) fine-tuning. OmniH2O [19] enables universal whole-body teleoperation and transfers the resulting policies to real hardware. GR00T-N1 [20], VLA [21] and ψ_0 [22] train generalist VLA policies on heterogeneous mixtures of robot demonstrations and human video. These approaches are fundamentally bottlenecked by the cost and scale of real-world data collection, limiting the diversity of scenes objects, and failure modes they can cover, and requiring post-training adaptation at the deployment scene.

Simulation on the other hand offers 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, with VIRAL [6] limiting their generalization to diverse real-world settings. ASAP [23] trains agile whole-body skills in simulation and refines them with real-world RL to close the sim-to-real gap, but focuses on a fixed set of motion skills rather than scene-level generalization.

C. Post-Training Policies with Reinforcement Learning

Reinforcement learning (RL) has become a standard tool for post-training policies initialized by imitation, pushing them past the demonstrator ceiling that pure behavior cloning cannot surpass. For diffusion and flow-based policies, DPPO [24] and ReinFlow [25] adapt policy-gradient updates to the multi-step denoising process, with ReinFlow converting the deterministic flow-matching ODE into a stochastic process amenable to such updates. Flow-GRPO [26] turns the ODE sampler into an SDE for GRPO-style optimization, while FPO reformulates importance sampling through the conditional flow-matching objective.

For humanoids specifically, HumanPlus [18] pairs behavior cloning from human demonstrations with RL fine-tuning, while ASAP [23] refine simulation-trained policies with *real-world* RL. In contrast, we post-train entirely in simulation across a diverse distribution of scenes, using Flow-GRPO to close the imitation gap left by scripted demonstrations without any real-world interaction.

III. VILMA- Data Generation

We generate all training data in simulation. Each episode samples a reach then pick-up task in a procedurally generated house and rolls it out with a scripted whole-body controller that solves the task from privileged simulator state, yielding observation-action trajectories directly usable for imitation (Fig. 2). Below we describe the policy interface the data is collected in (Section III-A), how tasks are sampled (Section III-B), the scripted demonstrator, and the per-episode domain randomization that diversifies the collected data.

A. Environments Setup

We use the Unitree G1 humanoid. At each policy step, $\pi_\theta(a_{t:t+H} \mid o_t)$ emits an action chunk of H 15-dimensional commands forwarded to the low-level controllers described below. The policy runs at $10\,\text{Hz}$ on 224×384 RGB images.

Observations. π_θ consumes $o_t = (I_t^{\text{head}}, I_t^{\text{wrist}}, o_t^{\text{prop}})$, where $I_t^{\text{head}}, I_t^{\text{wrist}} \in \mathbb{R}^{H \times W \times 3}$ are a head-mounted fisheye RGB image and a wrist-mounted RGB image, and the proprioceptive

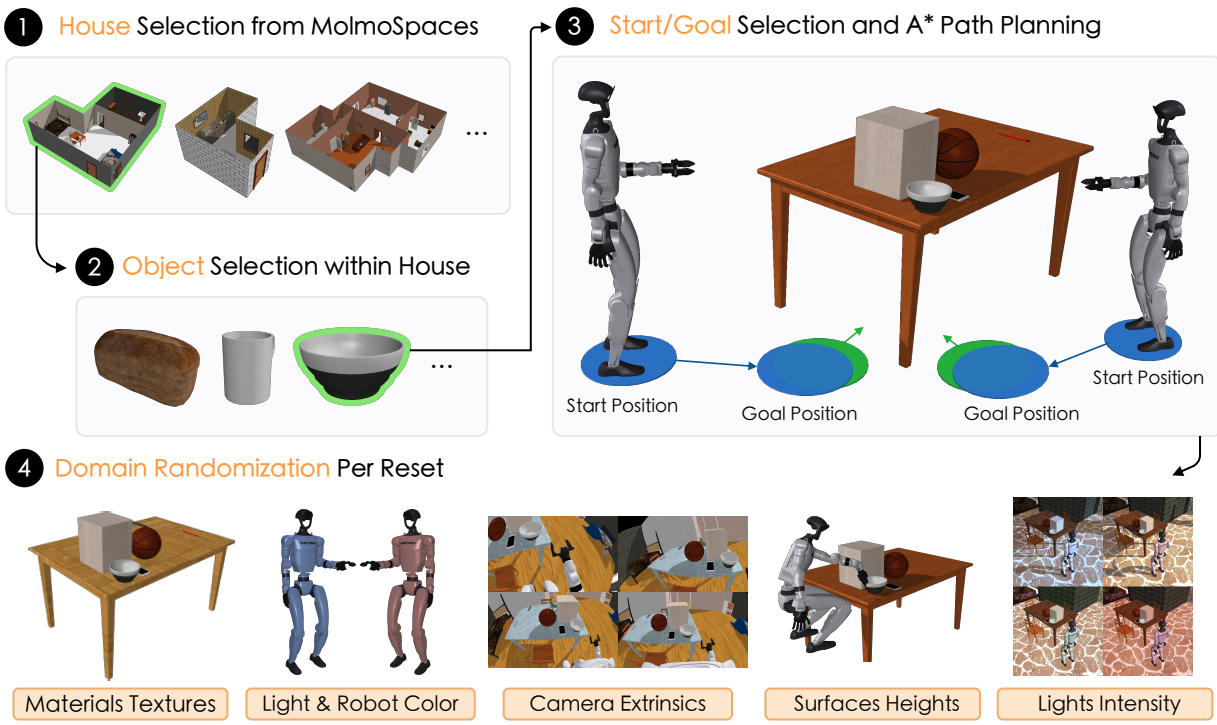

Fig. 2. Synthetic data generation pipeline. Each episode samples: (1) a house from MolmoSpaces; (2) a pickable target object within it; (3) a contact-free start and grasp-standoff goal pose connected by a planned path the scripted controller follows; and (4) domain randomization that is applied independently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

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.

Hierarchical Control. We adopt the action factorization standard in recent humanoid loco-manipulation systems [6, 7, 22]: π_θ runs at 10 Hz, while a pre-trained lower-body controller converts $(v_t, \bar{h}_t)$ into joint torques via a 50 Hz locomotion policy whose targets are PD-tracked at the 200 Hz physics rate; the upper-body and gripper commands $(\bar{q}_t^{\text{waist}}, \bar{q}_t^{\text{arm}}, \bar{g}_t)$ are PD-tracked at the same 200 Hz rate. Concretely, the lower-body controller is the decoupled balance-and-walking pair released with SONIC [27]: two separately trained policies dispatched per step by a hard threshold on the commanded base velocity ($\|v_t\| < 0.05$ m/s selects the balance policy). The same controller stack runs in simulation and on the real robot, so the command-to-behavior mapping is preserved across the sim-to-real gap. This decoupling matters in two concrete ways: (1) it shrinks the policy's output space from $\approx$ 29 joint torques to 15 semantically meaningful commands, which is what makes both behavior cloning and RL fine-tuning feasible at our scale; (2) the lower-body stack the policy is trained against at deployment time is bit-identical to the one used during training, so its closed-loop response to the policy's commands is a fixed property of the system rather than an additional sim-to-real shift.

B. Task Sampling

Episodes are instantiated from MolmoSpaces [4], an open ecosystem of procedurally generated indoor scenes populated with physically simulated objects. Our task sampler (Fig. 2) closely follows that of MolmoBot [5], which we extend to the humanoid setting. We describe the three components of the resulting joint distribution below.

Scene. A scene is drawn from a pool that spans the ProcTHOR-10k, Holodeck, and ProcTHOR-Objaverse splits curated by MolmoSpaces. Once a scene is 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 later both to reject infeasible base placements and to plan navigation waypoints.

Target object. Within the sampled scene, candidates are the free-jointed assets MolmoSpaces marks as *pickable*. One target instance is drawn per reset, alongside its associated set of grasps. Per-realization perturbations of the object's pose and supporting surface height are deferred to Section III-B2.

Robot sampling pose. Candidate base positions (x, y) are sampled from the free cells of $\mathcal{O}_s$ in an annulus $r_{\min} \leq \|p - p_{\text{obj}}\|_2 \leq r_{\max}$ 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 whose full whole-body pose is contact-free and from which the target is visible in the head fisheye camera. Crucially, we validate each candidate against the full simulator

rather than the inflated 2D occupancy grid. 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 the inner sampling radius 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.

1) Scripted Demonstrator: Each episode is rolled out by a scripted whole-body controller π_{ctrl} that consumes privileged simulator state (the target object's pose, a pre-sampled grasp transform, and the robot's base pose) and emits commands in the same 15-dimensional action interface used by the learned policy (Section III-A). Sharing the action space is what makes the resulting trajectories directly cloneable: only the mapping from observations to actions changes between π_{ctrl} and π_θ, not the underlying WBC.

Reaching. While the robot is far from the target, π_{ctrl} plans an A* path on an extra-inflated copy of the scene's free-space grid $\mathcal{O}_s$. The path is reduced to a short list of straight-line waypoints, and the controller 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} executes a discrete sequence of phases (reach, descend, close, lift), advancing to the next once the end-effector reaches the current phase's target. The grasp target itself is chosen per episode from the object's pre-computed grasp database (Section III-B): we transform each candidate into world coordinates using the sampled object pose and pick the first one that is reachable under Mink [28] 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 the controller can drop or raise its torso to bring the chosen grasp into the arm's dexterous workspace rather than forcing a single fixed standing height. Episodes for which no candidate satisfies these criteria, or in which any phase times out, are simply discarded.

2) Domain Randomization: On top of the task instance, every episode independently resamples the following axes, see Fig. 2.:

- **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; 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.

IV. VILMA- POLICY LEARNING

We train a single policy $\pi_\theta(a \mid o)$ in two stages. In **Stage 1** (Section IV-A), we behavior-clone the scripted controller π_{ctrl} of Section III-B1, giving π_θ a usable prior for a high-dimensional, partially observable control problem. In **Stage 2** (Section IV-B), we fine-tune π_θ with large-scale on-policy RL under a sparse task-success reward. This repairs the structural artifact π_{ctrl}'s discrete phase machine leaves on π_θ during behavior cloning, where the policy is forced 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 same stage doubles as a mechanism for instilling deployment behaviors that the imitation data cannot supervise, such as collision avoidance with scene geometry that the scripted controller never violates during data collection. The resulting checkpoint is deployed zero-shot on the real robot.

A. Imitation Learning Pretraining

We pretrain π_θ by behavior cloning on trajectories generated by the task sampler and scripted demonstrator described in Section III-B-Section III-B1. The data is collected under a fixed mixture: 80% *reach+pick* episodes that exercise the full navigation-to-grasp pipeline, and 20% *pick-only* episodes in which the robot is initialized at a grasp standoff and only the manipulation phase machine is executed. The pick-only episodes additionally perturb the robot's initial upper-body pose, sampling the arm joints around their defaults and drawing the initial gripper aperture uniformly between fully open and fully closed; the resulting trajectories teach the policy to recover from off-distribution starting configurations rather than the default arms-forward, gripper-open initial state seen in the reach+pick episodes. 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 [29], 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. This puts the regression targets near zero mean across the dataset and stabilizes flow-matching training. The policy is trained to predict action chunks of length 16; at deployment, only the first 8 actions of each chunk are executed before a new prediction is generated from the latest observation.

Fig. 3 summarizes our model's architecture. The vision encoder is a frozen DINOv3 ViT-B/16 [30] that emits a patch-token grid per camera; we concatenate the head- and wrist-camera token grids along the sequence axis. A two-layer MLP projects o_t^{prop} into a single state token that is appended to the visual context. The flow-matching head is a DiT-style transformer [31] that attends from H noised action tokens to the concatenated observation context via cross-attention.

We train the head with the rectified-flow matching loss [32]: for each chunk we sample noise $x_0 \sim \mathcal{N}(0, I)$ and a

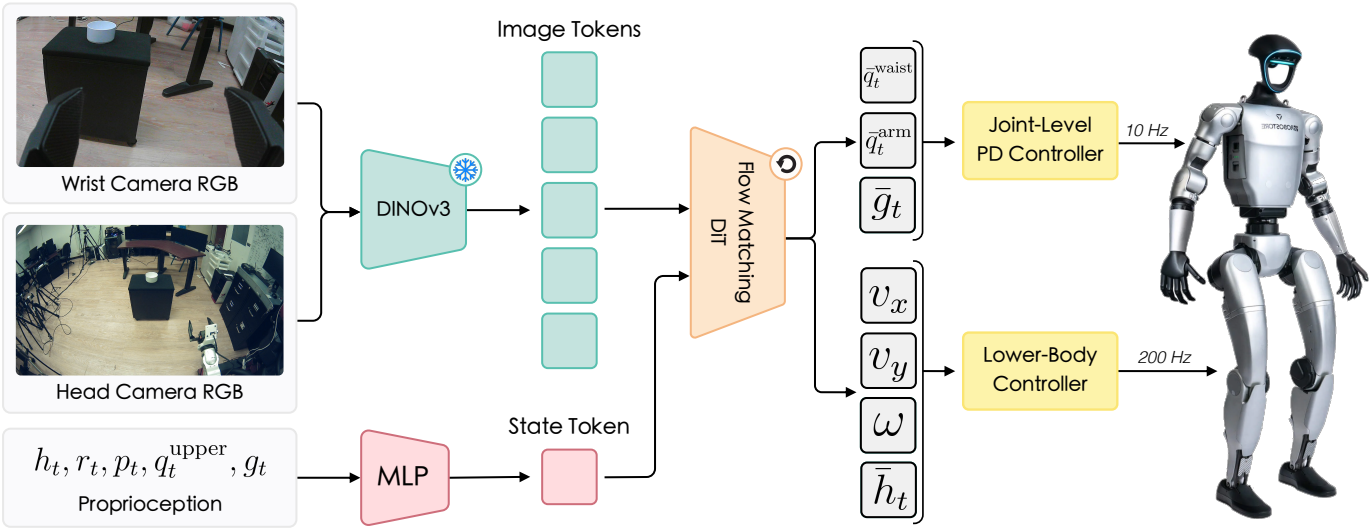

Fig. 3. Policy architecture. The head and wrist RGB streams are encoded by a frozen DINOv3 backbone into per-patch image tokens, while the proprioceptive vector $(h_t, r_t, p_t, q_t^{\text{upper}}, g_t)$ is mapped by an MLP to a single state token. A flow-matching DiT cross-attends to these tokens and 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.

timestep $\tau \in [0, 1]$ from a Beta-distributed schedule biased toward late timesteps (closer to the data), form $x_\tau = (1 - \tau)x_0 + \tau\, a_{t:t+H}$, and minimize the MSE between the predicted velocity $\hat{v}_\theta(x_\tau, \tau, o_t)$ and the target velocity $a_{t:t+H} - x_0$. At deployment, we initialize from noise and integrate the learned velocity field to recover the action chunk.

B. Reinforcement Learning Refinement

We refine π_θ from the imitation checkpoint with Flow-GRPO [26], 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.

The N parallel environments are partitioned into N/M groups of size M, each environment in a group reset to the same task and differing only in the SDE noise injected by π_θ. For a group G with returns $\{R_i\}_{i\in G}$, the advantage is the standardized group-relative return $A_i = (R_i - \mu_G)/\sigma_G$; no learned value function is used. The actor is updated with a PPO-style clipped objective [33] over the per-flow-step importance ratio, with dual-clipping to bound the loss from below on large negative-advantage samples. 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 grasp and lift 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.

V. Experiments

We evaluate the full recipe in simulation and in real to answer four research questions:

RQ1. Does our policy trained entirely in simulation transfer zero-shot to a real Unitree G1 robot?

RQ2. Which design choices, the vision encoder and the action parameterization, matter for transfer?

RQ3. How does behavior-cloning performance scale with the number of demonstrations?

RQ4. Does RL refinement close the imitation gap left by the scripted demonstrator?

A. Setup

Instantiating the data-generation pipeline of Section III, we collect 150k bowl-pick demonstrations, roughly 650 hours of robot experience across ∼50k distinct procedurally generated scenes, at ∼100 episodes per minute on a single NVIDIA L40S (∼40 GPU-hours in total). We behavior-clone π_θ on this set and, for the refined variant, further train it with Flow-GRPO as described in Section IV-B.

We measure two success rates throughout. Manipulation success rate (SR) isolates the manipulation phase: the robot is initialized at a grasp standoff and must grasp and lift the target. Loco-manipulation SR measures the full task, where the robot starts away from the object and must navigate to

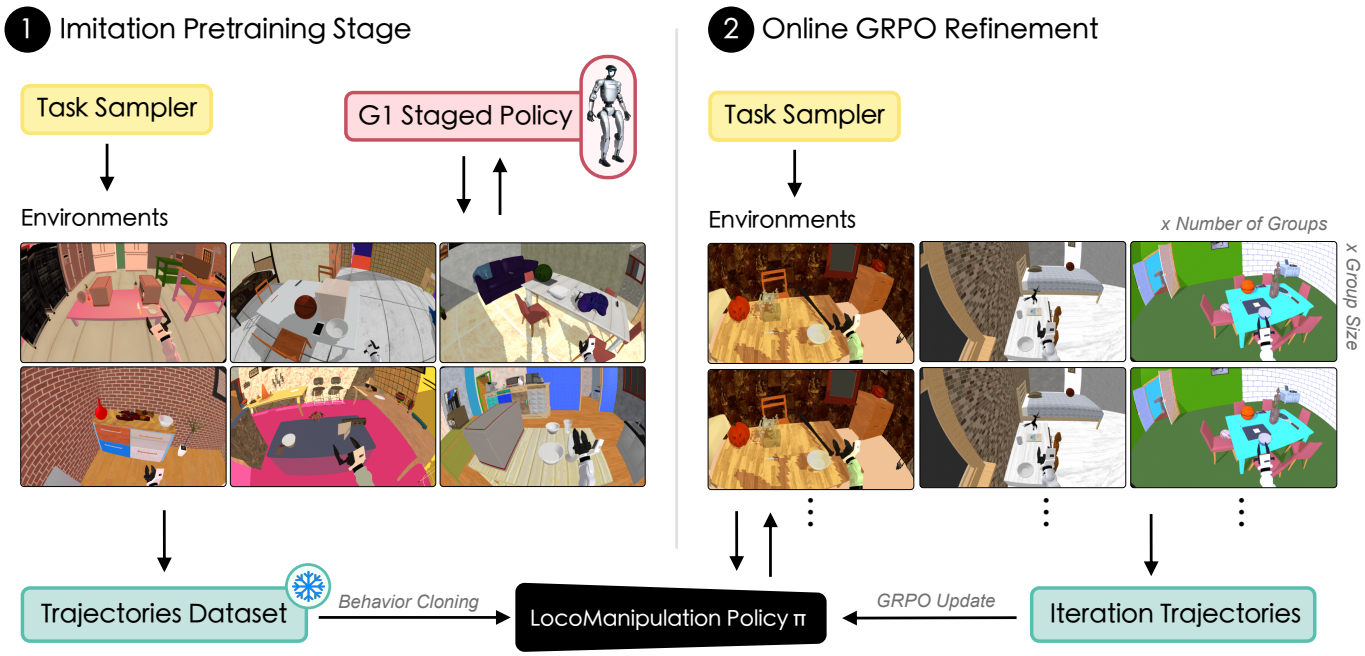

Fig. 4. Two-stage training pipeline. *Stage 1 (Imitation Pretraining):* the task sampler instantiates diverse simulated environments in which the scripted G1 staged policy generates demonstration trajectories; the resulting frozen dataset trains the loco-manipulation policy π by behavior cloning. *Stage 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.

Fig. 5. Real-world Rollouts. Representative zero-shot rollouts of ViLMa on the Unitree G1 across diverse, unseen scenes: the policy navigates to the target object and grasps it from head- and wrist-camera observations alone.

it before grasping, additionally stressing the locomotion-to-manipulation handoff that the scripted demonstrator advances through a hidden phase boundary. In simulation, we evaluate on a fixed set of 100 initializations never seen during training and report the best validation checkpoint (max-SR); the same checkpoint is deployed zero-shot on the physical G1 for 22 manipulation and 30 loco-manipulation trials per model. No real-world data is used at any stage.

B. Main Results

Table I reports our final model (Dino encoder, delta actions, 150k demos) before and after RL refinement, and Fig. 5 shows representative real-world rollouts. The policy transfers zero-shot to the physical G1 (RQ1), and RL lifts simulated loco-manipulation SR from 67% to 83% and real-world loco-manipulation SR from 56% to 73% (RQ4). Notably, real manipulation SR is nearly unchanged (73% vs. 77%): 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 IV predicts behavior cloning cannot recover from privileged-state-driven demonstrations. Standard errors are calculated assuming a binomial distribution.

	Simulation SR [%]		Real-World SR [%]	
Method	Manipulation	Loco-Manipulation	Manipulation	Loco-Manipulation
Num. Trials	100	100	22	30
ViLMa-BC	75 ± 4.3	67 ± 4.7	73 ± 9.5	56 ± 9.0
ViLMa (BC+RL)	$\mathbf{79 \pm 4.1}$	$\mathbf{83 \pm 3.8}$	$\mathbf{77 \pm 8.9}$	$\mathbf{73 \pm 8.1}$

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

C. Design Ablations

Table II ablates the two design choices that are significant for transfer (RQ2), holding everything else at the BC-150k configuration.

The vision encoder choice is decisive. A frozen DINOv3 encoder substantially outperforms SigLiP, in both simulation (67% vs. 42%) and the real world, where SigLiP collapses almost entirely (8% manipulation, 0% loco-manipulation). We attribute this to DINOv3's stronger geometric features, which transfer across the sim-to-real appearance gap without finetuning.

The action parameterization matters just as much. Delta actions, expressing absolute joint and pose targets relative to

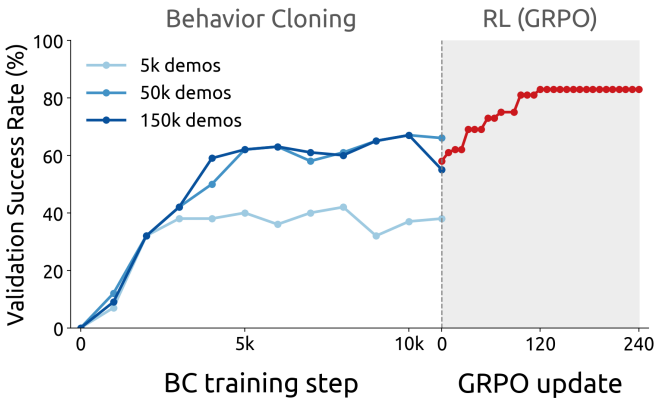

Fig. 6. Behavior cloning saturates; RL breaks through. Loco-manipulation success rate (%) on the 100-scene validation set across the full recipe. The left segment scales behavior-cloning demonstrations (5k–150k) and plateaus after 50k; the right segment continues the 150k checkpoint with GRPO, which pushes past the imitation ceiling.

a per-chunk anchor (Section IV-A), are critical for transfer. Absolute targets, trained for the same number of steps, reach only 17% manipulation and 0% loco-manipulation on hardware, confirming that the near-zero-mean delta targets are what make flow-matching imitation transfer reliably.

	Simulation SR [%]		Real-World SR [%]	
Variant	Manipulation	Loco-Manipulation	Manipulation	Loco-Manipulation
Num. Trials	100	100	12	10
ViLMa-BC (DINOv3, delta)	$\mathbf{75 \pm 4.3}$	$\mathbf{67 \pm 4.7}$	$\mathbf{73 \pm 12.5}$	$\mathbf{56 \pm 15.5}$
with SigLiP	58 ± 4.9	42 ± 4.9	8 ± 8.0	0
with Absolute Actions	62 ± 4.9	45 ± 5.0	17 ± 10.8	0

TABLE II. Architecture ablations. Success rates (%) $\pm$ standard error in simulation. BC policies from 150k demonstrations, DINOv3 features, and delta-action targets are each necessary for zero-shot transfer.

D. Data Scaling

Fig. 6 plots loco-manipulation SR across the entire training recipe on a single axis (RQ3): the left segment scales the number of behavior-cloning demonstrations (5k, 50k, 150k), and the right segment continues training the 150k checkpoint with GRPO. Behavior cloning improves steeply from 5k to 50k demonstrations (40% to 67%) and then *saturates*, with the further jump to 150k yielding no measurable gain (67%). This plateau is the empirical signature of the imitation ceiling argued in Section IV: because the scripted demonstrator's phase boundaries are not observable from the policy's inputs, additional demonstrations from the same demonstrator cannot teach the policy what it structurally cannot see. The GRPO segment then breaks through the plateau, raising loco-manipulation SR to 83%, confirming that the remaining headroom is recovered by a different learning signal rather than by more demonstrations.

VI. Conclusion

We presented ViLMa, 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 tens of thousands 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 GRPO recovers the remaining headroom in both simulation and the real world.

Limitations point to future work. Our deployed policy scale is bounded by compute rather than by the pipeline, which already generates data across tens of thousands of objects. Scaling this data to train a language-conditioned, object-generalist VLA is a natural next step. The sparse reward could also be enriched to instill behaviors such as collision avoidance, and the fixed lower-body controller could be adapted jointly with manipulation.

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*. IEEE, 2012, pp. 5026–5033.

[2] K. Zakka, Q. Liao, B. Yi, L. L. Lay, K. Sreenath, and P. Abbeel, "mjlab: 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. Zurbrügg, 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. Eftekhar, 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. [Online]. Available: 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 *Advances in Neural Information Processing Systems*,

M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 19 134–19 146. [Online]. Available: 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.*, "Deepseekmath: 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 *Proceedings of the 1st Annual Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, S. Levine, V. Vanhoucke, and K. Goldberg, Eds., vol. 78. PMLR, 13–15 Nov 2017, pp. 1–16. [Online]. Available: 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] OpenAI, I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, J. Schneider, N. Tezak, J. Tworek, P. Welinder, L. Weng, Q. Yuan, W. Zaremba, and L. Zhang, "Solving rubik's cube with a robot hand," *arXiv preprint arXiv:1910.07113*, 2019.

[14] 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.

[15] 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 *Proceedings of The 9th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Lim, S. Song, and H.-W. Park, Eds., vol. 305. PMLR, 27–30 Sep 2025, pp. 1004–1029. [Online]. Available: https://proceedings.mlr.press/v305/deng25a.html

[16] 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 *Proceedings of The 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, K. Liu, D. Kulic, and J. Ichnowski, Eds., vol. 205. PMLR, 14–18 Dec 2023, pp. 80–93. [Online]. Available: https://proceedings.mlr.press/v205/li23a.html

[17] 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.

[18] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," *arXiv preprint arXiv:2406.10454*, 2024.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] A. Ren, J. Lidard, L. Ankile, A. Simeonov, P. Agrawal, A. Majumdar, B. Burchfiel, H. Dai, and M. Simchowitz, "Diffusion policy policy optimization," in *International Conference on Learning Representations*, vol. 2025, 2025, pp. 77 288–77 329.

[25] 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*, vol. 38, pp. 106 282–106 319, 2026.

[26] 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*, vol. 38, pp. 40 783–40 818, 2026.

[27] 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.

[28] K. Zakka, "Mink: Python inverse kinematics based on mujoco," *URL https://github. com/kevinzakka/mink*,

vol. 10, p. 2, 2024.

[29] 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.

[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*, 2023, pp. 4195–4205.

[32] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

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