DreamControl: Human-Inspired Whole-Body Humanoid Control for Scene Interaction via Guided Diffusion

Dvij Kalaria[1,2] Sudarshan Harithas[1,3] Pushkal Katara[1] Sangkyung Kwak[1] Sarthak Bhagat[1] S. Shankar Sastry[2]
Srinath Sridhar[3] Sai Vemprala[1] Ashish Kapoor[1] Jonathan Huang[1]

***Abstract*— We introduce DreamControl, a novel methodology for learning autonomous whole-body humanoid skills. DreamControl leverages the strengths of diffusion models and Reinforcement Learning (RL): our core innovation is the use of a diffusion prior trained on human motion data, which subsequently guides an RL policy in simulation to complete specific tasks of interest (e.g., opening a drawer or picking up an object). We demonstrate that this human motion-informed prior allows RL to discover solutions unattainable by direct RL, and that diffusion models inherently promote natural-looking motions, aiding in sim-to-real transfer. We validate DreamControl's effectiveness on a Unitree G1 robot across a diverse set of challenging tasks involving simultaneous lower and upper body control and object interaction.**

Project website: https://genrobo.github.io/DreamControl/

I. Introduction

Significant advancements in humanoid robot control have been made in recent years, particularly in locomotion and motion tracking, leading to impressive demonstrations such as robot dancing [1], [2] and kung-fu [3]. However, for humanoid robots to transition from mere exhibitions to universal assistants, they must be able to interact with their environment by fully leveraging their humanoid form factor's mobility and extensive range of motion. This includes tasks such as stooping to pick up objects, squatting for heavy boxes, bracing to open drawers or doors, and precise pushing, punching, or kicking of specific targets.

These tasks are sometimes referred to as whole-body manipulation and loco-manipulation tasks, and continue to pose substantial challenges for the humanoid robotics field. Existing approaches to humanoid manipulation often simplify the problem by fixing the lower body (e.g., [4]), training upper and lower bodies separately with the lower body reacting to the upper (e.g., [5]), or focusing exclusively on computer graphics applications (e.g., [6], [7]).

A major challenge in whole-body loco-manipulation is that of contending with multiple timescales. First, there is the problem of dynamically maintaining stability and balance, which requires short-horizon control and robustness at the sub-second scale and is challenging due to high degrees of freedom, underactuation, and a high center of mass. Recent approaches address this part of the problem with reinforcement learning (RL) and sim-to-real transfer.

[1]General Robotics; This work was performed while Dvij Kalaria and Sudarshan Harithas were at General Robotics.
[2]University of California, Berkeley
[3]Brown University

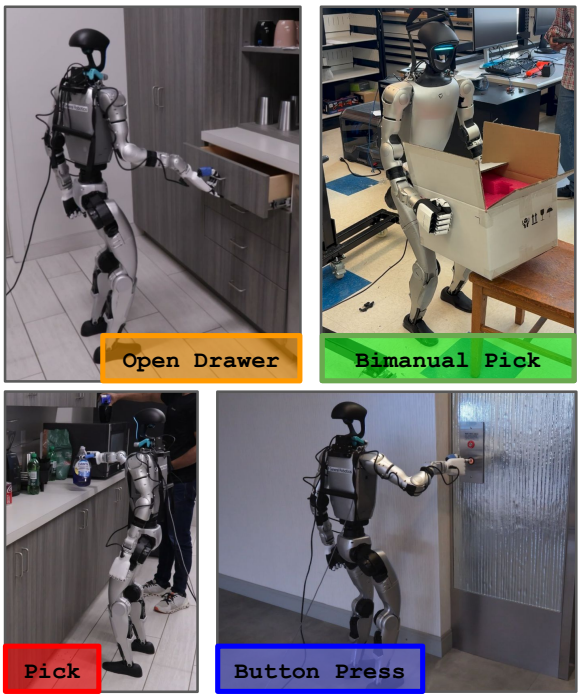

Fig. 1: Unitree G1 humanoid performing various skills trained via DreamControl, including (1) opening a drawer, (2) bimanual pick (of a box), (3) ordinary pick and (4) pressing an elevator button.

Concurrently, the robot needs to formulate a motion plan for grasping distant objects, which is a long-horizon problem, spanning up to tens of seconds. The long-horizon and high-dimensional nature of bimanual manipulation leads to a particularly challenging RL exploration problem, requiring complex and precise coordination between both sets of arms and hands. Directly applying RL in such scenarios can therefore often fail or lead to unnatural behaviors that generalize poorly to the real world [8].

Consequently, modern approaches often rely on real-world data collection and imitation learning. Among these approaches, diffusion policies [9] (and related flow matching based approaches [10]) have shown promise in generating long, consistent temporal data, offering a potential solution to these challenges. Conceptually, diffusion-based approaches are a natural fit for the multimodal nature of action distributions in manipulation and also scale well, allowing for learning multiple tasks simultaneously. A complication, however, is the limited availability of teleoperation data for whole-body humanoid control, leading some groups to propose using only upper-body teleoperation data. Whatever the form, however, collecting large teleoperation data can be

labor-intensive and difficult to scale (Goldberg [11] refers to this as the 100,000-year data gap in robotics).

We introduce *DreamControl*, a two-stage methodology for learning autonomous whole-body skills that explicitly addresses the above issues by leveraging the strengths of both diffusion models and RL. Our key innovation is the use of a diffusion prior over human motions, specifically utilizing OmniControl [12], which takes text conditions (e.g., "open the drawer") and spatiotemporal guidance (e.g., enforcing a wrist position at a specific time) as input. Subsequently, we retarget motion samples from this prior to the robot form factor of interest and train an RL policy in simulation to follow these retargeted samples while simultaneously completing some task of interest (e.g., lifting a heavy box). We demonstrate that both privileged and non-privileged versions of this policy can be trained with minor modifications, facilitating convenient deployment to real robots.

Our approach offers several benefits. First, instead of relying on teleoperation data, it only depends on human data for training the diffusion prior. Human motion data is far more abundant (e.g. from motion capture and video sources), and since it only informs the prior, we do not depend on access to explicit reference trajectories during policy rollouts, enabling fully autonomous task execution. We show that this prior enables RL to discover solutions unattainable by direct RL approaches. Additionally, our diffusion prior contributes to bridging the sim-to-real gap by proposing natural-looking (less robotic) motion plans that generally do not include extreme motions.

We demonstrate the success of DreamControl on a Unitree G1 robot across a variety of challenge tasks, including those emphasizing simultaneous lower and upper body control and object interaction, alongside ablations that validate our design choices.

II. Related work

Our work is inspired by three main strands of research: robot manipulation (with imitation learning as well as on-policy RL), RL for legged robots (from locomotion and teleoperation to full autonomy) as well as the character animation and human motion modeling literatures.

A. Recent Advances in Manipulation

Modern deep learning approaches to robot manipulation are commonly based on imitation learning [13]–[15]. Our work draws particularly on those that leverage diffusion [16], [17] or related flow matching [18] approaches to policy parameterization [9], [10], [19]–[23]. These approaches attempt to emulate the success of LLMs because they scale well given lots of data, but unlike text, robot data is not ubiquitously available on the internet. Collecting robot trajectories is costly, requiring expensive teleoperation rigs as well as training and paying human teleoperators.

There are also on-policy RL approaches trained in simulated environments that are more scalable [4], [24] — though robust sim2real transfer is challenging. Most relevant to our approach is the work of Lin et al [4] who demonstrate robust bimanual manipulation skills on a humanoid robot, but do not address whole body skills. Like [4] we use on-policy RL (instead of behavior cloning with teleoperated trajectories) — but our models are informed by a diffusion prior over human motion, significantly reducing the need for reward engineering.

B. RL controllers for legged robots

In recent years, deep RL has seen significantly increased adoption in RL controllers for legged robots, starting with robust legged locomotion policies for quadrupeds [25]–[27] followed by bipedal form factors (including humanoids) [28]–[34]. More recently authors have proposed whole body motion tracking and teleoperation approaches which allow a robot to track the motion of a human teleoperator [1], [2], [35]–[42] including advances in handling agile and extreme motions (e.g. KungFuBot [3] and ASAP [43]). See also [44] for a more complete overview of the field.

Finally, beyond tracking a provided human motion, lies the challenge of enabling fully autonomous execution of specific tasks, e.g. kicking, sitting, swinging a golf club (we will sometimes refer to these as "skills") [5], [43], [45]–[50].

Among these works, HumanPlus [48] and AMO [5] demonstrate whole body autonomous task execution but require teleoperated trajectories for IL. R2S2 [50] train a limited set of "primitive" skills and focus primarily on ensembling these primitives using IL and RL — whereas our focus is on a recipe for training a library of such primitive skills. Finally we note that BeyondMimic [49] also leverages both guided diffusion and RL, but the way that diffusion is used is mostly orthogonal to our work. Guidance in their diffusion policy is "coarse" rather than fine-grained compared to our work and does not account for object interaction or long range planning.

C. Character Animation and Motion Models

There is also a similar literature on modeling the movement of humanoids in physically realistic character animation settings [8], [51]–[58]. By having access to privileged simulation states and no sim-to-real distribution shifts, solving problems in this simplified synthetic setting first has proven useful as a stepping stone prior to crossing the sim-to-real gap.

We are in particular influenced by statistical priors over human motion — which have a rich history (see e.g. [59]–[61]) and today leverage the recent advances in generative AI (such as diffusion models and autogregressive transformers) [6], [7], [12], [62], [63].

Among these papers, our work is most influenced by OmniGrasp [8], CloSd [7] and TokensHSI [6], all of which explicitly handle object/scene interactions. Omnigrasp leverages a prior over human motions (PULSE, [55]) taking the form of a bottleneck VAE that directly predicts actions though has the disadvantage of being somewhat more awkward to interpret directly as a prior on human trajectories. CloSd generates motion plans via diffusion and using an RL-trained policy to execute in simulation. Our work goes further by leveraging

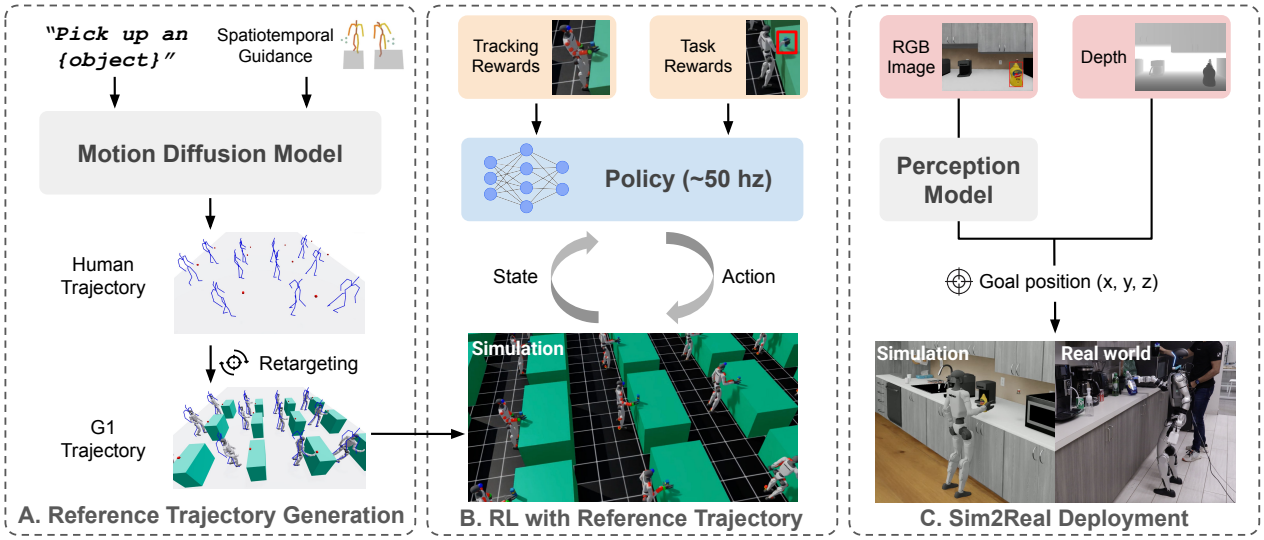

Fig. 2: **DreamControl Overview**: (A) we first generate text and spatiotemporally guided human motion trajectories using diffusion; (B) we train goal-conditioned RL policies to track these generated trajectories while completing some task of interest; (C) we deploy these policies to a real humanoid, leveraging off-the-shelf vision models to determine spatial guidance inputs for the RL policy.

richer/fine-grained guidance which allows us to handle a larger variety of tasks and addresses important sim2real aspects (such as removing explicit dependence on reference trajectories from a motion model), enabling deployment on a real robot.

III. The DreamControlMethodology for Constructing Humanoid Skills

Our approach starts conceptually with standard teleoperation RL pipelines for quadrupeds and bipeds / humanoids. Typically such an RL policy (e.g., [35], [36] is trained with a dense reward for accurately tracking keypoints from an input trajectory (obtained e.g., via motion capture of a human) along with other rewards (for stability, balance, smoothness, etc). However for training a humanoid to perform autonomous skills (such as picking up an object) an input trajectory is not available at test time so we need to either let RL learn the motion through exploration (which is very hard without careful reward engineering), or we need to generate this motion plan externally.

In DreamControl, we take this latter route of first generating motion plans externally through a pre-trained human motion prior. These generated motion plans are then used implicitly during RL training in the reward signal but not explicitly used as goal conditions by the policy, (hence putting the "Dream" in "DreamControl"). In addition to these dense tracking rewards, we also use sparse and verifiable task-specific rewards to explicitly promote task completion. The overall pipeline is summarized in Fig. 2. We now discuss the two stages of (1) trajectory generation from a human motion prior and (2) RL in more detail. More details about the exact parameters used for each task in these 2 stages are in the Appendix.

A. Stage 1: Generating Reference Trajectory from a Human Motion Prior

A key desiderata for our first stage is to leverage human motion data instead of humanoid teleoperation data which is expensive. Human motion data is widely available (either in the form of motion capture datasets or implicitly in video datasets) and thus allows us to learn high quality priors and a multitude of tasks. By generating realistic human-like motions we also hope for more seamless sim-to-real transfer and also offer more natural interaction with humans.

Additionally, we would like to choose a model that has favorable scaling properties with respect to data. We thus use a diffusion transformer [9], [62], [64] which has been shown to be successful in modeling human motion as well as robot manipulation trajectories and is known to scale well with large datasets while remaining well behaved in low data regimes [65].

Among previous motion diffusion models, we build on OmniControl [12] which can be flexibly conditioned on both text and spatiotemporal guidance. OmniControl generates trajectories following a given text command (e.g., "Pick up the bottle") while stipulating that a joint or a subset of joints reach a prespecified spatial location at a prespecified time. In this way, our trajectory generation stage is analogous to image or video inpainting.

This form of spatiotemporal guidance allows us to connect a generated trajectory to its environment (for example, allowing us to specify where a humanoid should sit, how high it should jump, or the location of the object to be manipulated). Being able to control this interaction point is critical, since in the RL simulator we can instantiate an object at some location and then use trajectories that are guaranteed to approach this object, significantly facilitating the RL exploration problem. In DreamControl, we specifically design the form of spatiotemporal guidance for each

task. For instance, our pick task involves providing a spatial target for the wrist. See the Appendix for more details on guidance control implementations.

Post-retargeting and trajectory filtering: Since Omnicontrol is trained on human trajectories (represented via the SMPL parameterization [66]), we next retarget these generated trajectories to the G1 form factor (in similar fashion to [67]): by solving an optimization problem (using the PyRoki [67] library) that minimizes the relative keypoint positions, relative angles, and a scale factor that adjusts for the difference in link lengths. Additional residuals, such as feet contact costs, self-collision costs, and foot orientation costs, are used to improve physical plausibility.

Finally, we apply a layer of post-processing to the generated G1 trajectories prior to passing on to RL. Some of the generated trajectories are not dynamically feasible and thus not fit to be used for tracking in Stage 2. We devise task-specific filtering mechanisms based on some heuristics as discussed in detail in the Appendix. We also apply task-specific trajectory refinements to avoid unnecessary movements, such as setting all left arm joints to a default value in the `Pick` task, where only right arm is used. These are also discussed in the Appendix.

Trajectory representation: After all post-filtering and refinement, we have a set of reference trajectories, $\{\alpha_i\}$ generated with the same task-specific text prompt with different $(p_{k,i},\, t_{k,i})$ spatiotemporal "goals", which mean that joint k should be at position $p_{k,i}$ at time $t_{k,i}$. In addition to this, we also define t_g as the time at which the task-specific goal interaction occurs. For example, t_g for the `Pick` task is the time when the object is to be picked up, t_g for the `Button Press` task is when the button is to be pressed, etc. This t_g is crucial for synthesizing scenes for each given reference trajectory, as we discuss later in Section III-B.

Each reference trajectory is represented as a sequence of target frames, $\alpha_i = [\alpha_{i,0},\, \alpha_{i,\Delta t},\, ...,\, \alpha_{i,(L-1)\Delta t}]$, where $\Delta t = 0.05s$ is the time step and $L = 196$ is trajectory length (hence each trajectory spans $9.8s$). Each frame is represented as $\alpha_t = \{p_t^{\text{ref,root}},\, \theta_t^{\text{ref,root}},\, q_t^{\text{ref}},\, s_t^{\text{ref,left}},\, s_t^{\text{ref,right}}\}$ where $q_t^{\text{ref}} \in \mathbb{R}^{27}$ are the reference joint angles, $p_t^{\text{ref,root}} \in \mathbb{R}^3$ is the position of the root, $\theta_t^{\text{ref,root}} \in \mathbb{R}^4$ is the orientation of the root in quaternions, and $s_t^{\text{ref,left}}, s_t^{\text{ref,right}} \in \{0, 1\}$ are the left and right hand states with 0 denoting open and 1 denoting closed. These are manually labeled for each task; for example, for the pick-with-right-hand task, we ensure that the right hand closes immediately after time t_g and that the left hand stays closed through the duration of the task. Refer to the Appendix for more details.

Out-of-distribution tasks: We use OmniControl in this paper in a "zero-shot" fashion in the sense that we use the weights and hyperparameters as originally released by authors, and retarget them to G1 after trajectory generation. Since OmniControl is trained on HumanML3d [68], we find that it is capable of handling a wide variety of tasks "out-of-the-box".

However, we also explore a method handling certain novel tasks that are not well represented by Omnicontrol training distribution (e.g., pulling drawers) by using IK-based optimization on a base trajectory of a person standing idle (or bending down to pull a drawer below waist-level). More details are described in the Appendix.

B. Stage 2: RL with Reference Trajectory

Once we have the reference trajectories from Stage 1, we formulate the interactive task as an RL problem. In this section, we describe a "privileged" variant with access to internal simulator states and defer our discussion of how to adapt this approach for real deployments to Section IV.

Scene synthesis: First we need to synthesize a scene that makes sense for each of the Stage 1 kinematic trajectories to execute the interactive task — for example if we had used guidance in Stage 1 to ask that the wrist be at point p at time t, then during RL we instantiate the object-to-be-manipulated near point p. More formally, given the time t_g at which the interaction happens in the generated trajectory (e.g., the time at which the object is picked, button is pressed etc.), we place the object of interest (pick object, button etc.) at the following location:

$$t^{\text{o,world}} = \mathbf{t}_{t_g}^{\text{b,world}} + R_{t_g}^{\text{b,world}} \mathbf{t}^{\text{o,b}}, \tag{1}$$

$$R^{\text{o,world}} = R_{t_g}^{\text{b,world}} R_{t_g}^{\text{o,b}}, \tag{2}$$

where $(\mathbf{t}_{t_g}^{\text{b,world}}, R_{t_g}^{\text{b,world}})$ is the pose of the robot body part link in question in world frame (e.g., the right wrist link for the pick-with-right-hand task) and $(\mathbf{t}^{\text{o,b}}, R^{\text{o,b}})$ is the offset of the object w.r.t the robot body part link where the object should be placed. All specific offsets and body part links used in each task are listed in the Appendix. We randomize the timestamp t_g, target positions used to generate trajectories, and thus p_{t_g}, and other characteristics of the object such as mass and friction. We choose randomization hyperparameters to demonstrate a wide range of settings for which we can generate and use reference trajectories to solve tasks that can be scaled with careful engineering. The exact randomization hyperparameters of the environment are reported in the Appendix.

Action space: Next we define our action and observation spaces. Our simulated robot is a 27-DoF Unitree G1 equipped with two 7-DoF DEX 3-1 hands, one mounted on each wrist (in real world experiments we use Inspire hands). In this work we restrict hand control to discrete open/closed configurations that are fixed per-task (see Appendix for task-specific settings, e.g., extending the right index finger for the open configuration during the button-press task). The action space is therefore defined as $a_t \in \mathbb{R}^{29}$, where $a_t = \{a_t^{\text{body}}, a_t^{\text{left}}, a_t^{\text{right}}\}$, with $a_t^{\text{body}} \in \mathbb{R}^{27}$ denoting the target joint angles for the G1 body, and $a_t^{\text{left}}, a_t^{\text{right}} \in \mathbb{R}$ controlling the left and right hands, respectively. For the hand controls, negative values correspond to an open hand and positive values to a closed hand.

Observations: For each task, we include proprioception information (joint angles $q_t^{\text{robot}} \in \mathbb{R}^{27}$, joint velocities $\dot{q}_t^{\text{robot}} \in \mathbb{R}^{27}$), root linear velocity $v_t^{\text{root}} \in \mathbb{R}^3$, root angular velocity $\omega_t^{\text{root}} \in \mathbb{R}^3$, projected gravity in root frame $g_t \in \mathbb{R}^3$, previous

TABLE I: Reward terms for reference tracking and smooth policy enforcement.

Reward Term	Interpretation
$\|q_t^{\text{robot}} - q_t^{\text{ref}}\|_2$	Penalizes deviation from reference joint angles
$\|p_t^{\text{ref,key}} - p_{t,\text{robot}}^{\text{key}}\|_2$	Penalizes deviation from reference keypoints (3D positions in world frame)
$\|p_t^{\text{robot,root}} - p_t^{\text{ref,root}}\|$	Penalizes deviation of robot root from reference root position
$\|\theta_t^{\text{rel}}\|$	Penalizes deviation in orientation between robot and reference
$\|\sigma(a_t^{\text{left}}) - s^{\text{ref, left}}\| + \|\sigma(a_t^{\text{right}}) - s_t^{\text{ref, right}}\|$	Penalizes deviation of hand states from reference ($\sigma(x) = \frac{1}{1+e^{-x}}$)
$\|\tau_t\|_2 + \|\ddot{q}_t^{\text{robot}}\|_2$	Penalizes high torques and accelerations
$\|\frac{a_t - a_{t-1}}{\Delta t}\|_2$	Penalizes high action rate changes
$\sum_{k=\{\text{left foot,right foot}\}} \|c^{\text{f}}(pos_t^{\text{robot, f}} - pos_{t-1}^{\text{robot, f}})\|$	Penalizes foot sliding while in ground contact
n^{feet}	Penalizes excessive foot-ground contacts (to discourage baby steps)
$\theta_z^{\text{left foot,z}} + \theta_z^{\text{right foot,z}}$	Encourages feet to remain parallel to the ground (discourages heel sliding)

action $a_{t-1} \in \mathbb{R}^{29}$, and a target trajectory reference as input observation, along with privileged task-specific observations like relative pose of the object, mass, friction of the object wherever relevant. At time t, the target trajectory reference observation consists of $[\gamma_t, \gamma_{t+\Delta t^{\text{obs}}}, ..., \gamma_{t+(K-1)\Delta t^{\text{obs}}}]$ where K is the number of time steps into the future, and $\Delta t^{\text{obs}} = 0.1s$ is a hyperparameter. γ_t consists of $(q_t^{\text{ref}}, \dot{q}_t^{\text{ref}}, p_t^{\text{rel, root}}, p_t^{\text{rel, key}}, s_t^{\text{ref, left}}, s_t^{\text{ref, right}})$ where $q_t^{\text{ref}} \in \mathbb{R}^{27}$ are the target joint angles, $\dot{q}_t^{\text{ref}} \in \mathbb{R}^{27}$ are the target joint velocities, $p_t^{\text{rel,root}}$ is the relative pose of the root reference with respect to the robot's base, $p_t^{\text{rel, key}} \in \mathbb{R}^{3\times 41}$ corresponds to the relative position of the 41 keypoints on the robot with respect to it's root and $s_t^{\text{ref,left}}, s_t^{\text{ref,right}}$ are the target reference binary hand states. Note that γ_t essentially contains the same information as α_t but is transformed into the robot's frame and some redundant information is added for ease of policy learning inspired from [36], [40], [43]. Unlike these other works, we include relative pose $p_t^{\text{rel, root}}$ as input instead of root reference velocities, and target reference keypoints of reference with respect to robot's root instead of target trajectory's root. This is because works like [40] do not aim to precisely track the trajectory but instead train a deployable policy that aims to follow velocity commands of root, and thus tend to drift from global reference trajectory. In our work, as we aim to precisely follow trajectories to accomplish interactive tasks, we exploit the privileged sim global root position to obtain relative keypoints as observation. However, it must be noted that it should be possible to later train a non-privileged vision-based policy that exploits scene information from vision to implicitly replace the global position privileged information.

Rewards: In Table I, we summarize our reward terms (1) for tracking the reference correctly, (2) to encourage maintaining balance and enable smooth control.

We also add some (3) task-specific rewards to encourage accomplishing the task with high success rates, such as the reward for raising an object above a height for the pick task. These are described for each task in the Appendix. The total reward r_t is obtained by:

$$r_t = \Sigma_{i=1}^{10} w_{r_i} r_{t,i} + w_{r^{\text{task,sparse}}} r^{\text{task,sparse}},$$

where w_{r_i} for $i \in \{1, 2, ..., 10\}$ (indexing over our 10 reward terms) and $w_{r^{\text{task,sparse}}}$ are task-specific weights whose exact values are given in the Appendix.

TABLE II: Success rates (%) in simulation over 1000 random environments. (a) *TaskOnly*; (b) *TaskOnly+*; (c) *TrackingOnly*. **Bold** denotes the best results.

Task / Method	(a)	(b)	(c)	Ours
`Pick`	0	15.1	87.5	**95.4**
`Bimanual Pick`	0	31.0	**100**	**100**
`Pick from Ground (Side Grasp)`	0	0	99.4	**100**
`Pick from Ground (Top Grasp)`	0	0	**100**	**100**
`Press Button`	0	**99.8**	99.1	99.3
`Open Drawer`	0	24.5	**100**	**100**
`Open Door`	0	15.4	**100**	**100**
`Precise Punch`	0	**100**	99.4	99.7
`Precise Kick`	0	97.6	96.1	**98.6**
`Jump`	0	0	**100**	**100**
`Sit`	0	**100**	**100**	**100**

Training: We setup our environment and training in IsaacLab [69] that uses IsaacSim simulation. All policies are trained on an NVIDIA RTX A6000 with 48 GB vRAM using PPO [70]. For each task, we train for 2000 iterations with 8192 parallel environments. See Appendix for more details.

IV. Evaluation

A. Tasks and Baselines

We evaluate on a library of 11 tasks: `Pick`, `Bimanual Pick`, `Pick from Ground (Side Grasp)`, `Pick from Ground (Top Grasp)`, `Press Button`, `Open Drawer`, `Open Door`, `Precise Punch`, `Precise Kick`, `Jump`, and `Sit`. For comparison, we report results for our method (DreamControl), which combines tracking with task-specific sparse rewards, and three baselines:

(a) *TaskOnly*: only task-specific (sparse) rewards,
(b) *TaskOnly+*: only task-specific rewards, both sparse and engineered dense rewards inspired by [8], and
(c) *TrackingOnly*: only tracking rewards.

B. Simulation Results

Table II reports success rates over 1000 random environments, with each task success criteria defined in the Appendix. Our results show that *TaskOnly* (a) achieves 0% success across all tasks, since relying solely on sparse rewards provides no dense guidance to discover meaningful motions. *TaskOnly+* (b) improves performance by adding engineered dense terms, enabling success on simpler tasks

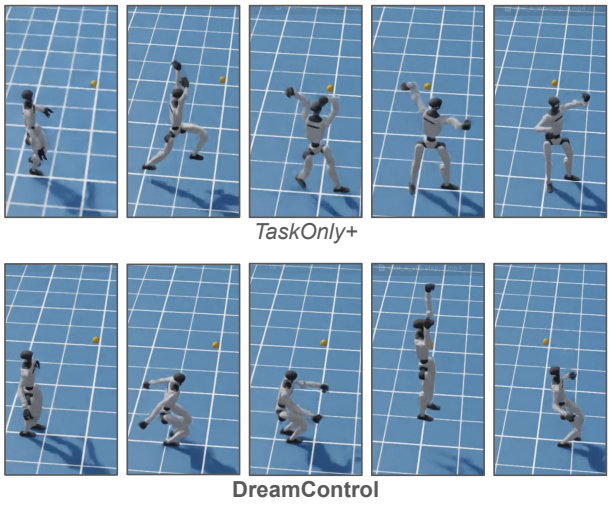

Fig. 3: Comparison of trajectories for the task of `Jump`. The top row shows results from the *TaskOnly+* baseline, while the bottom row illustrates trajectories from the DreamControl. The yellow sphere depicts the spatial control point used to guide the trajectories.

like `Press Button` and `Precise Punch`, but still fails on tasks requiring coordinated whole-body motion. For example, in `Pick from Ground`, the robot must crouch in a balanced manner, and in `Jump` it must first lower its body before springing upward; with only a pelvis-target reward, the policy instead "settles for" merely stretching its knees but is unable to discover how to perform a real jump (see Fig. 3). *TrackingOnly* (c) performs better overall, but struggles with fine-grained interactive tasks such as `Pick`. By combining both tracking and task-specific signals, DreamControl achieves robust performance, outperforming all baselines and achieving the best results on 9 of 11 tasks.

C. Human-ness Comparison

We also evaluate the human-ness (or naturalness) of trajectories generated by our policies compared to *TaskOnly+*. For tasks where *TaskOnly+* achieves non-zero success, we assess how natural the resulting motions appear. First, we report Fréchet inception distance (FID; [71]) scores on the HumanML3D dataset [68], using task-matched ground-truth trajectories obtained via keyword filtering (see Appendix). As shown in Table III, DreamControl consistently achieves lower FID values than *TaskOnly+*, indicating closer alignment with human motions. An exception occurs in the `Pick` task, which we conjecture is due to a domain gap: human demonstrations typically involve waist-level picking, whereas the shorter G1 robot often performs shoulder-level picks, making its trajectories less comparable to the human dataset. However, other metrics suggest that our method DreamControlproduces more human-like motions consistently in all tasks.

To further assess motion quality, we calculate average absolute jerk (Table III, second-to-last column), defined as $\frac{\sum_i \sum_t \sum_k |\dddot{p}^{\cdot,\text{key,global}}_{t,k}|}{NTK}$, where $\dddot{p}^{\cdot,\text{key,global}}_{t}$ is the third derivative of the global position of the k^{th} keypoint, with $N = 1000$ trajectories, $T = 500$ time-steps, and $K = 41$ keypoints.

TABLE III: Human-ness comparison of DreamControl (Ours) and TaskOnly+. We report FID and jerk (m/s^3), where lower is better, and the average human preference. **Bold** denotes the best results.

Task	Method	FID ↓	Jerk ↓	User Study ↑
`Pick`	*TaskOnly+*	**0.240**	211.2	15.0%
	Ours	0.320	**147.5**	**85.0%**
`Press Button`	*TaskOnly+*	1.220	235.7	17.25%
	Ours	**0.375**	**161.9**	**82.75%**
`Precise Punch`	*TaskOnly+*	0.417	229.9	7.5%
	Ours	**0.084**	**199.8**	**92.5%**
`Precise Kick`	*TaskOnly+*	0.522	360.9	17.5%
	Ours	**0.161**	**252.5**	**82.5%**
`Jump`	*TaskOnly+*	1.216	236.4	5.0%
	Ours	**0.208**	**148.5**	**95.0%**

Lower jerk indicates smoother motions, and our method significantly outperforms *TaskOnly+*, producing more fluid, human-like movements.

We also conduct a user study with 40 participants, in which they are shown side-by-side videos of trajectories from both methods (order randomized) and asked to select which looked more human-like. As summarized in Table III (last column), participants overwhelmingly preferred DreamControl's trajectories across all tasks, further confirming the naturalness of our approach.

Finally, as shown in Fig. 3, the trajectory generated by our approach (bottom row) is noticeably more natural and human-like compared to the *TaskOnly+* baseline (top row). Our method exhibits a smooth jumping motion, where the robot first bends and then lifts off the ground, while the *TaskOnly+* baseline lifts off but without bending, resulting in a less human-like motion that also does not accomplish the task.

D. Sim2Real Deployment

To demonstrate real-world effectiveness, we deploy policies for selected tasks on hardware after retraining with observations modified in the following ways to remove dependence on simulator-privileged information:

- Remove the trajectory reference observation (ref), though references remain available via the rewards;
- Remove the linear velocity of the root;
- Remove privileged scene-physics information like object mass, friction etc.;
- Add time encoding $(t, \sin(2\pi t/T))$, where T is the total length of the episode.

We use the same rewards as in Stage 2, including motion tracking terms, but transform the reference trajectory for (x, y, yaw) of the root to avoid privileged inputs that are unavailable for the critic. The resulting policy depends only on the relative position of the object / goal, making it deployable on the real robot.

Hardware setup: We use a Unitree G1 humanoid (27-DoF, waist lock mode allowing only yaw movement) equipped with Inspire dexterous hands (6-DoF each, controlled in binary open / close mode). An onboard IMU provides root

orientation, gravity direction, and angular velocity. A RealSense D435i depth camera, mounted on the neck, estimates the 3D position of the object / goal relative to the pelvis.

Deployment: To estimate object positions, we leverage an off-the-shelf open-vocabulary object detection model, OWLv2 [72] for 2D localization, then lift to 3D using depth and object-specific offsets (Fig. 2(C)). Due to OWLv2's inference latency, we detect the object only in the first frame and hold the estimate fixed thereafter. To mitigate errors from this static estimate, we freeze the lower body during interactive tasks (except bimanual pick) and add a penalty on root velocities to ensure the base remains static. Note that this limitation stems from perception bottlenecks rather than our method; in principle, vision-based policies could be trained via student-teacher distillation (e.g., [4]), which we leave for future work. We provide details for sim2real transfer in the Appendix.

We successfully deployed policies for: `Pick` (standing), `Bimanual Pick` (with boxes of varying weights), `Press Button` (standing), `Open Drawer` (at different positions), `Precise Punch` (standing) and `Squat` (with varying depths). Representative visualizations are shown in Fig. 1 and more videos are available at Link.

V. Discussion and Future Work

In this work, we presented DreamControl, a novel recipe for training autonomous humanoid skills that leverages guided diffusion for long-horizon planning and reinforcement learning for robust control, without requiring expensive demonstration data. We validated our approach on several challenging tasks, successfully transferring policies from simulation to a real G1 humanoid robot.

While our current implementation does not yet compose skills, support dexterous manipulation or complex object geometries, the data-efficient nature of our method provides a strong foundation for extensions along these lines. We believe that scaling DreamControl to a broader repertoire of tasks and more diverse robot morphologies is a promising and immediate next step toward more capable and general-purpose humanoid robots.

ACKNOWLEDGMENT

We are thankful for discussions and help with experiments from B. Rishi, Y. Patel, D. Narayanan, S. Deolasee, V. Rajesh, G. Moffatt.

References

[1] X. Cheng, Y. Ji, J. Chen, R. Yang, G. Yang, and X. Wang, "Expressive whole-body control for humanoid robots," *arXiv preprint arXiv:2402.16796*, 2024.

[2] M. Ji, X. Peng, F. Liu, J. Li, G. Yang, X. Cheng, and X. Wang, "Exbody2: Advanced expressive humanoid whole-body control," *arXiv preprint arXiv:2412.13196*, 2024.

[3] W. Xie, J. Han, J. Zheng, H. Li, X. Liu, J. Shi, W. Zhang, C. Bai, and X. Li, "Kungfubot: Physics-based humanoid whole-body control for learning highly-dynamic skills," *arXiv preprint arXiv:2506.12851*, 2025.

[4] T. Lin, K. Sachdev, L. Fan, J. Malik, and Y. Zhu, "Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids," *arXiv preprint arXiv:2502.20396*, 2025.

[5] J. Li, X. Cheng, T. Huang, S. Yang, R. Qiu, and X. Wang, "Amo: Adaptive motion optimization for hyper-dexterous humanoid whole-body control," *Robotics: Science and Systems 2025*, 2025.

[6] L. Pan, Z. Yang, Z. Dou, W. Wang, B. Huang, B. Dai, T. Komura, and J. Wang, "Tokenhsi: Unified synthesis of physical human-scene interactions through task tokenization," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5379–5391.

[7] G. Tevet, S. Raab, S. Cohan, D. Reda, Z. Luo, X. B. Peng, A. H. Bermano, and M. van de Panne, "CLoSD: Closing the loop between simulation and diffusion for multi-task character control," in *The Thirteenth International Conference on Learning Representations*, 2025.

[8] Z. Luo, J. Cao, S. Christen, A. Winkler, K. Kitani, and W. Xu, "Omnigrasp: Grasping diverse objects with simulated humanoids," *Advances in Neural Information Processing Systems*, vol. 37, pp. 2161–2184, 2024.

[9] 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*, p. 02783649241273668, 2023.

[10] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, *et al.*, "π0: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550," *arXiv preprint ARXIV.2410.24164*.

[11] K. Goldberg, "Good old-fashioned engineering can close the 100,000-year "data gap" in robotics," p. eaea7390, 2025.

[12] Y. Xie, V. Jampani, L. Zhong, D. Sun, and H. Jiang, "Omnicontrol: Control any joint at any time for human motion generation," in *The Twelfth International Conference on Learning Representations*, 2024.

[13] Q. Vuong, S. Levine, H. R. Walke, K. Pertsch, A. Singh, R. Doshi, C. Xu, J. Luo, L. Tan, D. Shah, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models," in *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[14] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6892–6903.

[15] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, *et al.*, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[16] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*. pmlr, 2015, pp. 2256–2265.

[17] D. Watson, J. Ho, M. Norouzi, and W. Chan, "Learning to efficiently sample from diffusion probabilistic models," *arXiv preprint arXiv:2106.03802*, 2021.

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

[19] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, *et al.*, "Octo: An open-source generalist robot policy," *arXiv preprint arXiv:2405.12213*, 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] 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," *arXiv preprint arXiv:2507.05331*, 2025.

[22] J. Wen, Y. Zhu, J. Li, Z. Tang, C. Shen, and F. Feng, "Dexvla: Vision-language model with plug-in diffusion expert for general robot control," *arXiv preprint arXiv:2502.05855*, 2025.

[23] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, "Rdt-1b: a diffusion foundation model for bimanual manipulation," *arXiv preprint arXiv:2410.07864*, 2024.

[24] K. Li, P. Li, T. Liu, Y. Li, and S. Huang, "Maniptrans: Efficient dexterous bimanual manipulation transfer via residual learning," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 6991–7003.

[25] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," 2021.

[26] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning (CoRL)*, 2023.

[27] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots," *arXiv preprint arXiv:2309.14341*, 2023.

[28] A. Kumar, Z. Li, J. Zeng, D. Pathak, K. Sreenath, and J. Malik, "Adapting rapid motor adaptation for bipedal robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 1161–1168.

[29] Z. Zhuang, S. Yao, and H. Zhao, "Humanoid parkour learning," in *8th Annual Conference on Robot Learning*, 2024.

[30] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Real-world humanoid locomotion with reinforcement learning," *Science Robotics*, vol. 9, no. 89, p. eadi9579, 2024.

[31] B. van Marum, A. Shrestha, H. Duan, P. Dugar, J. Dao, and A. Fern, "Revisiting reward design and evaluation for robust humanoid standing and walking," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 11 256–11 263.

[32] Y. Seo, C. Sferrazza, H. Geng, M. Nauman, Z.-H. Yin, and P. Abbeel, "Fasttd3: Simple, fast, and capable reinforcement learning for humanoid control," *arXiv preprint arXiv:2505.22642*, 2025.

[33] J. He, C. Zhang, F. Jenelten, R. Grandia, M. BÄcher, and M. Hutter, "Attention-based map encoding for learning generalized legged locomotion," *arXiv preprint arXiv:2506.09588*, 2025.

[34] A. Allshire, H. Choi, J. Zhang, D. McAllister, A. Zhang, C. M. Kim, T. Darrell, P. Abbeel, J. Malik, and A. Kanazawa, "Visual imitation enables contextual humanoid control," *arXiv preprint arXiv:2505.03729*, 2025.

[35] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi, "Learning human-to-humanoid real-time whole-body teleoperation," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 8944–8951.

[36] 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.

[37] Y. Ze, Z. Chen, J. P. AraÃšjo, Z.-a. Cao, X. B. Peng, J. Wu, and C. K. Liu, "Twist: Teleoperated whole-body imitation system," *arXiv preprint arXiv:2505.02833*, 2025.

[38] T. He, W. Xiao, T. Lin, Z. Luo, Z. Xu, Z. Jiang, J. Kautz, C. Liu, G. Shi, X. Wang, *et al.*, "Hover: Versatile neural whole-body controller for humanoid robots," *arXiv preprint arXiv:2410.21229*, 2024.

[39] P. Dugar, A. Shrestha, F. Yu, B. van Marum, and A. Fern, "Learning multi-modal whole-body control for real-world humanoid robots," 2024.

[40] Z. Chen, M. Ji, X. Cheng, X. Peng, X. B. Peng, and X. Wang, "Gmt: General motion tracking for humanoid whole-body control," *arXiv preprint arXiv:2506.14770*, 2025.

[41] Y. Li, Y. Lin, J. Cui, T. Liu, W. Liang, Y. Zhu, and S. Huang, "Clone: Closed-loop whole-body humanoid teleoperation for long-horizon tasks," *arXiv preprint arXiv:2506.08931*, 2025.

[42] Y. Zhang, Y. Yuan, P. Gurunath, T. He, S. Omidshafiei, A.-a. Aghamohammadi, M. Vazquez-Chanlatte, L. Pedersen, and G. Shi, "Falcon: Learning force-adaptive humanoid loco-manipulation," *arXiv preprint arXiv:2505.06776*, 2025.

[43] 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.

[44] Z. Gu, J. Li, W. Shen, W. Yu, Z. Xie, S. McCrory, X. Cheng, A. Shamsah, R. Griffin, C. K. Liu, *et al.*, "Humanoid locomotion and manipulation: Current progress and challenges in control, planning, and learning," *arXiv preprint arXiv:2501.02116*, 2025.

[45] T. Zhang, B. Zheng, R. Nai, Y. Hu, Y.-J. Wang, G. Chen, F. Lin, J. Li, C. Hong, K. Sreenath, *et al.*, "Hub: Learning extreme humanoid balance," *arXiv preprint arXiv:2505.07294*, 2025.

[46] J. Mao, S. Zhao, S. Song, T. Shi, J. Ye, M. Zhang, H. Geng, J. Malik, V. Guizilini, and Y. Wang, "Learning from massive human videos for universal humanoid pose control," *arXiv preprint arXiv:2412.14172*, 2024.

[47] H. Xue, X. Huang, D. Niu, Q. Liao, T. Kragerud, J. T. Gravdahl, X. B. Peng, G. Shi, T. Darrell, K. Screenath, *et al.*, "Leverb: Humanoid whole-body control with latent vision-language instruction," *arXiv preprint arXiv:2506.13751*, 2025.

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

[49] Q. Liao, T. E. Truong, X. Huang, G. Tevet, K. Sreenath, and C. K. Liu, "Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion," *arXiv e-prints*, pp. arXiv–2508, 2025.

[50] Z. Zhang, C. Chen, H. Xue, J. Wang, S. Liang, Y. Liu, Z. Zhang, H. Wang, and L. Yi, "Unleashing humanoid reaching potential via real-world-ready skill space," 2025.

[51] 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 (TOG)*, vol. 37, no. 4, pp. 1–14, 2018.

[52] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.

[53] X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler, "Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters," *ACM Transactions On Graphics (TOG)*, vol. 41, no. 4, pp. 1–17, 2022.

[54] Z. Luo, J. Cao, K. Kitani, W. Xu, *et al.*, "Perpetual humanoid control for real-time simulated avatars," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 10 895–10 904.

[55] Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu, "Universal humanoid motion representations for physics-based control," *arXiv preprint arXiv:2310.04582*, 2023.

[56] Y. Wang, Q. Zhao, R. Yu, H. W. Tsui, A. Zeng, J. Lin, Z. Luo, J. Yu, X. Li, Q. Chen, *et al.*, "Skillmimic: Learning basketball interaction skills from demonstrations," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 17 540–17 549.

[57] Z. Luo, J. Wang, K. Liu, H. Zhang, C. Tessler, J. Wang, Y. Yuan, J. Cao, Z. Lin, F. Wang, *et al.*, "Smplolympics: Sports environments for physically simulated humanoids," *arXiv preprint arXiv:2407.00187*, 2024.

[58] A. Tirinzoni, A. Touati, J. Farebrother, M. Guzek, A. Kanervisto, Y. Xu, A. Lazaric, and M. Pirotta, "Zero-shot whole-body humanoid control via behavioral foundation models," *arXiv preprint arXiv:2504.11054*, 2025.

[59] M. Brand and A. Hertzmann, "Style machines," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000, pp. 183–192.

[60] Y. Li, T. Wang, and H.-Y. Shum, "Motion texture: a two-level statistical model for character motion synthesis," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 465–472.

[61] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 283–298, 2007.

[62] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-or, and A. H. Bermano, "Human motion diffusion model," in *The Eleventh International Conference on Learning Representations*, 2023.

[63] S. Xu, H. Y. Ling, Y.-X. Wang, and L. Gui, "Intermimic: Towards universal whole-body control for physics-based human-object interactions," in *CVPR*, 2025.

[64] 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.

[65] M. Prabhudesai, M. Wu, A. Zadeh, K. Fragkiadaki, and D. Pathak, "Diffusion beats autoregressive in data-constrained settings," *arXiv preprint arXiv:2507.15857*, 2025.

[66] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," in *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 2023, pp. 851–866.

[67] C. M. Kim*, B. Yi*, H. Choi, Y. Ma, K. Goldberg, and A. Kanazawa, "Pyroki: A modular toolkit for robot kinematic optimization," 2025.

[68] C. Guo, S. Zou, X. Zuo, S. Wang, W. Ji, X. Li, and L. Cheng, "Generating diverse and natural 3d human motions from text," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 5152–5161.

[69] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg, "Orbit: A unified simulation framework for

interactive robot learning environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3740–3747, 2023.

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

[71] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, 2017.

[72] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," *Advances in Neural Information Processing Systems*, vol. 36, pp. 72 983–73 007, 2023.

[73] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *ArXiv*, vol. abs/1710.06542, 2017.

APPENDIX

VI. ROBOT DETAILS

A. Joints

Our Unitree G1 Edu+ consists of 27 joints with their names as mentioned in Table IV.

TABLE IV: Joints of the Unitree G1 Edu+ grouped by body part.

Legs	
left_hip_pitch_joint	right_hip_pitch_joint
left_hip_roll_joint	right_hip_roll_joint
left_hip_yaw_joint	right_hip_yaw_joint
left_knee_joint	right_knee_joint
left_ankle_pitch_joint	right_ankle_pitch_joint
left_ankle_roll_joint	right_ankle_roll_joint
Waist	
waist_yaw_joint	
(Left | Right) Arms	
left_shoulder_pitch_joint	right_shoulder_pitch_joint
left_shoulder_roll_joint	right_shoulder_roll_joint
left_shoulder_yaw_joint	right_shoulder_yaw_joint
left_elbow_joint	right_elbow_joint
left_wrist_roll_joint	right_wrist_roll_joint
left_wrist_pitch_joint	right_wrist_pitch_joint
left_wrist_yaw_joint	right_wrist_yaw_joint
(Left | Right) Hands	
left_hand_index_0_joint	right_hand_index_0_joint
left_hand_index_1_joint	right_hand_index_1_joint
left_hand_middle_0_joint	right_hand_middle_0_joint
left_hand_middle_1_joint	right_hand_middle_1_joint
left_hand_thumb_0_joint	right_hand_thumb_0_joint
left_hand_thumb_1_joint	right_hand_thumb_1_joint
left_hand_thumb_2_joint	right_hand_thumb_2_joint

B. Keypoints

The 41 keypoints on the robot are named as detailed in Table V.

C. PD control

We use PD controller to convert target joint angles to torque, τ_t as follows:-

$$\tau_t = k_p(q_t^{\text{commands}} - q_t^{\text{robot}}) - k_d\dot{q}_t^{\text{robot}}$$

Where k_p and k_d are positional and derivative gains and q_t^{commands} are the joint commands. The k_p, k_d gains for each joint are given in Table VI

VII. OPEN AND CLOSED STATES FOR EACH TASK

Each task has a specific set of joint angles for open and closed hand states to facilitate task-specific function. For instance, we want all fingers to open and closed in open and closed states of hand respectively while for button pressing, we only need the index finger open. For boxing, we keep all fingers closed for both open and closed as we never need to open them for the task. The exact finger joint angles for open and closed hand states for each task are listed in Table VII.

TABLE V: Keypoints of the Unitree G1 Edu+ grouped by body part.

Legs	
left_hip_pitch_link	right_hip_pitch_link
left_hip_roll_link	right_hip_roll_link
left_hip_yaw_link	right_hip_yaw_link
left_knee_link	right_knee_link
left_ankle_pitch_link	right_ankle_pitch_link
left_ankle_roll_link	right_ankle_roll_link
Waist & Torso	
pelvis	pelvis_contour_link
waist_yaw_link	waist_roll_link
torso_link	waist_support_link
logo_link	
Head & Sensors	
head_link	imu_link
d435_link	mid360_link
Arms	
left_shoulder_pitch_link	right_shoulder_pitch_link
left_shoulder_roll_link	right_shoulder_roll_link
left_shoulder_yaw_link	right_shoulder_yaw_link
left_elbow_link	right_elbow_link
left_wrist_roll_link	right_wrist_roll_link
left_wrist_pitch_link	right_wrist_pitch_link
left_wrist_yaw_link	right_wrist_yaw_link
left_rubber_hand	right_rubber_hand

VIII. KEYWORD-BASED FILTERING FOR FID

We use the following keywords to filter out trajectories that are used for FID calculation to evaluate human-ness of keypoint trajectories followed by **Ours** against *TaskOnly+*:-

- `Pick`: All motions whose prompts contain "pick" keyword in them
- `Press Button`: All motions whose prompts contain "press" keyword and "button" keyword in them
- `Precise Punch`: All motions whose prompts contain "punch" keyword in them
- `Precise Kick`: All motions whose prompts contain "kick" keyword in them
- `Jump`: All motions whose prompts contain "jump" keyword in them

IX. REFERENCE TRAJECTORY GENERATION

For each task, we have a text prompt, λ^{text} and a spatial control signal $\lambda^{\text{spatial}} \in \mathcal{R}^{L*S*3}$ where $L = 196$ is the no of time-step, $S = 22$ is the number of SMPL joints named as follows:-

TABLE VI: Joint list (unrolled column-wise) with default angle, K_p, and K_d all initialized to 0.

Joint name	**Default angle**	K_p	K_d
left_hip_pitch_joint	-0.2	200	5
left_hip_roll_joint	0	150	5
left_hip_yaw_joint	0	150	5
left_knee_joint	0.42	200	5
left_ankle_pitch_joint	-0.23	20	2
left_ankle_roll_joint	0	20	2
right_hip_pitch_joint	-0.2	200	5
right_hip_roll_joint	0	150	5
right_hip_yaw_joint	0	150	5
right_knee_joint	0.42	200	5
right_ankle_pitch_joint	-0.23	20	2
right_ankle_roll_joint	0	20	2
waist_yaw_joint	0	200	5
left_shoulder_pitch_joint	0.35	40	10
left_shoulder_roll_joint	0.16	40	10
left_shoulder_yaw_joint	0	40	10
left_elbow_joint	0.87	40	10
left_wrist_roll_joint	0	40	10
left_wrist_pitch_joint	0	40	10
left_wrist_yaw_joint	0	40	10
left_hand_index_0_joint	0	5	1.25
left_hand_index_1_joint	0	5	1.25
left_hand_middle_0_joint	0	5	1.25
left_hand_middle_1_joint	0	5	1.25
left_hand_thumb_0_joint	0	5	1.25
left_hand_thumb_1_joint	0	5	1.25
left_hand_thumb_2_joint	0	5	1.25
right_shoulder_pitch_joint	0.35	40	10
right_shoulder_roll_joint	-0.16	40	10
right_shoulder_yaw_joint	0	40	10
right_elbow_joint	0.87	40	10
right_wrist_roll_joint	0	40	10
right_wrist_pitch_joint	0	40	10
right_wrist_yaw_joint	0	40	10
right_hand_index_0_joint	0	5	1.25
right_hand_index_1_joint	0	5	1.25
right_hand_middle_0_joint	0	5	1.25
right_hand_middle_1_joint	0	5	1.25
right_hand_thumb_0_joint	0	5	1.25
right_hand_thumb_1_joint	0	5	1.25
right_hand_thumb_2_joint	0	5	1.25

TABLE VIII: Body joints grouped by body part.

Legs	
left_hip	right_hip
left_knee	right_knee
left_ankle	right_ankle
left_foot	right_foot
Spine & Torso	
pelvis	spine_1
spine_2	spine_3
neck	head
Arms	
left_collar	right_collar
left_shoulder	right_shoulder
left_elbow	right_elbow
left_wrist	right_wrist

For a given spatial control signal point at a given time-step and joint, it is considered functional only if the 3D spatial point is not $(0, 0, 0)$. Hence, we initiate all spatial control points with $(0, 0, 0)$ and then fill out values of joints at time-steps that we want to control.

TABLE VII: Open and Closed hand states for all tasks. Each config state for a hand consists of a tuple of 7 joint angles in the same order as in Table VIII. AOL: $(0, 0, 0, 0, 0, 0, 0)$, ACL: $(-\pi/2, -\pi/2, -\pi/2, -\pi/2, 0, \pi/3, \pi/2)$, AOR: $(0, 0, 0, 0, 0, 0, 0)$, ACR: $(\pi/2, \pi/2, \pi/2, \pi/2, 0, \pi/3, \pi/2)$, BPR: $(0, 0, \pi/2, \pi/2, 0, \pi/3, \pi/2)$, DOR: $(0, \pi/2, 0, \pi/2, 0, 0, 0)$

Task	Left Hand Config (Open — Close)	Right Hand Config (Open — Close)	Group set to default q
`Pick`	ACL — ACL	AOR — ACR	$G^{\text{left arm}}$
`Precise Punch`	ACL — ACL	ACR — ACR	-
`Precise Kick`	ACL — ACL	ACR — ACR	-
`Press Button`	ACL — ACL	PBR — PBR	-
`Jump`	ACL — ACL	ACR — ACR	-
`Sit`	ACL — ACL	ACR — ACR	-
`Bimanual Pick`	ACL — ACL	ACR — ACR	-
`Pick` from Ground (side grasp)	AOL — ACL	ACR — ACR	$G^{\text{right arm}}$
`Pick` from Ground (top grasp)	ACL — ACL	ACR — ACR	$G^{\text{left arm}}$
`Pick and Place`	ACL — ACL	AOR — ACR	-
`Open Drawer`	ACL — ACL	DOR — ACR	-
`Open Door`	ACL — ACL	DOR — ACR	-

A. *Task-specific prompts*

All task-specific prompt texts, λ^{text} are listed in Table IX. The spatial control signals for each task are given as follows:-

1) `Pick`*:* For each reference trajectory:- We sample a target point, $p^{\text{target}} = \{p^x \in \mathcal{U}(1.0, 1.2), p^y \in \mathcal{U}(-0.4, 0.4), p^z = 1.1\}$. Target time-step is chosen to be, $t'_g = 50 + \lfloor 50(p^x - 1.0) \rfloor$. Then, we set spatial control signal for wrist as follows:-

$$\lambda_i^{\text{right\_wrist}} = p^{\text{target}} \forall i \in \{t'_g, ..., t'_g + 20\}$$
$$\lambda_i^{\text{right\_wrist}} = (p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}} + 0.2) \quad \forall i \in \{t'_g + 20, ..., t'_g + 40\}$$

We also set the target spatial points for elbow to encourage generating trajectories where the object is grabbed from side as follows:-

$$\lambda_i^{\text{elbow}} = (p^{\text{target,x}}, p^{\text{target,y}} - 0.26\cos(\frac{\pi}{4}), p^{\text{target,z}} - 0.26\sin(\frac{\pi}{4})) \quad \forall i \in \{t'_g, ..., t'_g + 20\}$$

2) `Precise Punch`*:* For each reference trajectory:- We sample a target point, $p^{\text{target}} = \{p^x \in \mathcal{U}(1.2, 1.5), p^y \in \mathcal{U}(-0.2, 0.0), p^z = \mathcal{U}(1.0, 1.5)\}$. Target time-step is chosen to be, $t'_g = 30$. Then, we set spatial control signal for wrist as follows:-

$$\lambda_i^{\text{right\_wrist}} = p^{\text{target}} \forall i \in \{t'_g - 10, ..., t'_g + 10\}$$

3) `Precise Kick`*:* For each reference trajectory:- We sample a target point, $p^{\text{target}} = \{p^x \in \mathcal{U}(1.0, 1.2), p^y = 0.0, p^z = \mathcal{U}(0.5, 1.0)\}$. Target time-step is chosen to be, $t'_g = 30$. Then, we set spatial control signal for right foot as

TABLE IX: Text prompts for each task in simulation

Task	Prompts
`Pick`	"a person walks to cup, grabs the cup from side and lifts up"
`Precise Punch`	"a person performs a single boxing punch with his right hand"
`Precise Kick`	"a person stands and kicks with his right leg"
`Press Button`	"a person walks towards elevator, presses elevator button"
`Jump`	"a person jumps forward"
`Sit`	"a person walks towards a chair, sits down"
`Bimanual Pick`	"a person raises the toolbox with both hands"
`Pick from Ground (Side Grasp)`	"a person raises the toolbox with the use of one hand"
`Pick` from. Ground (Top Grasp)	"a person walks forward, bends down to pick something up off the ground"
`Pick and Place`	"a person picks the cup and puts it on another table"

TABLE X: Text prompts and slow-down factor for each task deployed on real robot

Task	Prompts	Slow down factor
`Pick`	"a person stands in place, grabs the cup from side and lifts up"	2.5
`Precise Punch`	"a person performs a single boxing punch with his right hand"	1.5
`Press Button`	"a person stands in place, presses elevator button"	1.5
`Bimanual Pick`	"a person raises the toolbox with both hands"	1
`Squat`	"a person squats in place and stands up"	1

follows:-

$$\lambda_i^{\text{right\_foot}} = p^{\text{target}} \forall i \in \{t'_g - 13, ..., t'_g + 10\}$$

4) `Press Button`: For each reference trajectory:- We sample a target point, $p^{\text{target}} = \{p^x \in \mathcal{U}(1.4, 1.8), p^y = \mathcal{U}(-0.4, 0.4), p^z = \mathcal{U}(1.1, 1.2)\}$. Target time-step is chosen to be, $t'_g = 70 + \lfloor 50(p^x - 1.4) \rfloor$. Then, we set spatial control signal for right foot as follows:-

$$\lambda_i^{\text{right\_wrist}} = p^{\text{target}} \forall i \in \{t'_g, ..., t'_g + 20\}$$
$$\lambda_i^{\text{right\_wrist}} = p^{\text{target,x}} - 0.2, p^{\text{target,y}}, p^{\text{target,z}} \text{ for } i = t'_g + 40$$

5) `Jump`: We sample a trajectory with target point, $p^{\text{target}} = \{p^x = 1.0, p^y = 0.0, p^z = 1.9\}$. Target time-step is chosen to be $t'_g = 50$. We set the spatial control signal for pelvis as: $\lambda_i^{\text{pelvis}} = p^{\text{target}}$ for $i = t'_g$

6) `Sit`: We sample a trajectory with target point, $p^{\text{target}} = \{p^x = 1.3, p^y = 0.8, p^z = 0.58\}$. Target time-step is chosen to be $t'_g = 100$. We set the spatial control signal for pelvis as: $\lambda_i^{\text{pelvis}} = p^{\text{target}}$ for $i = t'_g$

7) `Bimanual Pick`: We use target point, $p^{\text{target}} = \{p^x = 0.7, p^y = 0.0, p^z = 0.65\}$. Target time-step is chosen to be $t'_g = 98$. We set the spatial control signal for left wrist as follows:-

$$\lambda_i^{\text{left\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}} + 0.35, p^{\text{target,z}} + 0.25 - \frac{0.25i}{t'_g}\} \forall i \in \{0, ..., t'_g\}$$
$$\lambda_i^{\text{left\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}} + 0.15, p^{\text{target,z}} - 0.25 + \frac{0.25i}{t'_g}\} \forall i \in \{t'_g, ..., t'_g + 98\}$$

Spatial control signal for right wrist is as follows:-

$$\lambda_i^{\text{right\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}} - 0.35, p^{\text{target,z}} + 0.25 - \frac{0.25i}{t'_g}\} \forall i \in \{0, ..., t'_g\}$$
$$\lambda_i^{\text{right\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}} - 0.15, p^{\text{target,z}} - 0.25 + \frac{0.25i}{t'_g}\} \forall i \in \{t'_g, ..., t'_g + 98\}$$

8) `Pick from Ground (Side Grasp)`: We use target point, $p^{\text{target}} = \{p^x = 0.5, p^y = 0.5, p^z = 0.1\}$. Target time-step is $t'_g = 98$. We set the spatial control signal for the left wrist as follows:-

$$\lambda_i^{\text{left\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}} + 0.5 - \frac{0.5i}{t'_g}\} \forall i \in \{0, ..., t'_g\}$$
$$\lambda_i^{\text{left\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}} - 0.5 + \frac{0.5i}{t'_g}\} \forall i \in \{t'_g, ..., t'_g + 98\}$$

9) `Pick from Ground (Top Grasp)`: We use target point, $p^{\text{target}} = \{p^x = 1.0, p^y = 0.0, p^z = 0.2\}$. Target time-step is $t'_g = 50$. We set the spatial control signal for the right wrist as follows:-

$$\lambda_i^{\text{right\_wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}}\} \text{ for } i = t'_g$$

10) `Pick and Place`: We use 2 target points, $p^{\text{target,1}} = \{p^x = 1.2, p^y = -0.15, p^z = 0.6\}$ and $p^{\text{target,2}} = \{p^x = 1.2, p^y = -0.6, p^z = 0.6\}$. Target time-steps are $t'_{g1} = 80$

and $t'_{g2} = 160$. We set the spatial control signal for the right wrist as follows:-

$$\lambda_i^{\text{right\_wrist}} = p^{\text{target,1}} \text{ for } i = t'_{g1}$$
$$\lambda_i^{\text{right\_wrist}} = p^{\text{target,2}} \text{ for } i = t'_{g2}$$

11) `Open Drawer`: For drawer opening, the number of trajectories in HumanML dataset are very limited (only 18 with both "drawer" and "open" keywords in text description), due to which the generated motion was very off when prompted with drawer opening and only giving sparse spatial control signal for wrist at the drawer. To address this, we prompt DreamControl to generate 2 trajectories and no spatial control signal, one with prompt "stand still" and one with prompt "squat and stay in squat position". We use the generated motion as initialization and solve for the following optimization problem with gradient descent from the current state. The target points are sampled as $p^{\text{target}} = \{p^x \in \mathcal{U}(0.3, 0.35), p^y \in \mathcal{U}(-0.2, -0.1), p^z \in \mathcal{U}(0.4, 0.8)\}$ and $t'_g = 40$ if $p^z \geq 0.7$ and $t'_g = 50$ if $p^z < 0.7$.

We define target trajectory, τ^{wrist} for wrist as follows if $p^z > 0.7$:-

$$\tau_i^{\text{wrist}} = \{(p^{\text{target,x}}) * \frac{i}{40}, -0.25 + (p^{\text{target,y}} + 0.25) * \frac{i}{40}, 0.7 + (p^{\text{target,z}} - 0.7) * quad(\frac{i}{40})\}$$
$$\text{where } quad(x) = 1 - (x-1)^2, \forall i \in \{0, 1...40\}$$
$$\tau_i^{\text{wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{41, 42...70\}$$
$$\tau_i^{\text{wrist}} = \{p^{\text{target,x}} - \frac{0.2(i-70)}{40}, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{71, 72...110\}$$
$$\tau_i^{\text{wrist}} = \{p^{\text{target,x}} - 0.2, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{111, 112...196\}$$

and for $p^z < 0.7$ where w_i is the wrist position in the original re-targeted trajectory.

$$\tau_i^{\text{wrist}} = w_i \forall i \in \{0, 1, ..., 50\}$$
$$\tau_{i+50}^{\text{wrist}} = \{w_{50}^x + (p^{\text{target,x}} - w_{50}^x) * \frac{i}{40}, w_{50}^y + (p^{\text{target,y}} - w_{50}^y) * \frac{i}{40}, w_{50}^z + (p^{\text{target,z}} - w_{50}^z) * quad(\frac{i}{40})\}$$
$$\text{where } quad(x) = 1 - (x-1)^2, \forall i \in \{0, 1, ..., 40\}$$
$$\tau_{i+50}^{\text{wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{41, 42...70\}$$
$$\tau_{i+50}^{\text{wrist}} = \{p^{\text{target,x}} - \frac{0.2(i-70)}{40}, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{71, 72, ..., 110\}$$
$$\tau_{i+50}^{\text{wrist}} = \{p^{\text{target,x}} - 0.2, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{111, 112, ..., 146\}$$

With this target trajectory, we run gradient descent to optimize only the joint angles q_i^{ref} with the following loss in order to align the wrist to the target trajectory:-

$$L = \Sigma_{i=0}^{i=L-1} (p_i^{\text{wrist}} - w_i)^2$$
$$\text{where } p_i^{\text{wrist}} = fk^{\text{wrist}}(p_i^{\text{ref,root}}, \theta_i^{\text{ref,root}}, q_i^{\text{ref},*})$$
$$q_i^{\text{ref,j+1}} = q_i^{\text{ref,j}} - \lambda^{\text{lr}} \frac{\partial}{\partial q_i^{\text{ref},*}} L$$

12) `Open Door`: We generate trajectories for door opening in a similar way to drawer opening with the only changes in the target trajectory as follows for $p^{\text{target,z}} \geq 0.7$:-

$$\tau_i^{\text{wrist}} = \{(p^{\text{target,x}}) * \frac{i}{40}, -0.25 + (p^{\text{target,y}} + 0.25) * \frac{i}{40}, 0.7 + (p^{\text{target,z}} - 0.7) * quad(\frac{i}{40})\}$$
$$\text{where } quad(x) = 1 - (x-1)^2, \forall i \in \{0, 1...40\}$$
$$\tau_i^{\text{wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{41, 42...70\}$$
$$\tau_i^{\text{wrist}} = \{c^x - R\sin(a_i) - h\cos(a_i), c^y + R\cos(a_i) - h\sin(a_i), p^{\text{target,z}}\}$$
$$\forall i \in \{71, 72...110\} \text{ where } c^x = p^{\text{target,x}} + h, c^y = p^{\text{target,y}} - R, a_i = \frac{i-70}{40}$$
$$\tau_i^{\text{wrist}} = \{p^{\text{target,x}} - 0.2, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{111, 112...196\}$$

and for $p^{\text{target,z}} < 0.7$:-

$$\tau_i^{\text{wrist}} = w_i \forall i \in \{0, 1, ..., 50\}$$
$$\tau_{i+50}^{\text{wrist}} = \{w_{50}^x + (p^{\text{target,x}} - w_{50}^x) * \frac{i}{40}, w_{50}^y + (p^{\text{target,y}} - w_{50}^y) * \frac{i}{40}, w_{50}^z + (p^{\text{target,z}} - w_{50}^z) * quad(\frac{i}{40})\}$$
$$\text{where } quad(x) = 1 - (x-1)^2, \forall i \in \{0, 1, ..., 40\}$$
$$\tau_{i+50}^{\text{wrist}} = \{p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{41, 42...70\}$$
$$\tau_{i+50}^{\text{wrist}} = \{c^x - R\sin(a_i) - h\cos(a_i), c^y + R\cos(a_i) - h\sin(a_i), p^{\text{target,z}}\}$$
$$\forall i \in \{71, 72...110\} \text{ where } c^x = p^{\text{target,x}} + h, c^y = p^{\text{target,y}} - R, a_i = \frac{i-70}{40}$$
$$\tau_{i+50}^{\text{wrist}} = \{p^{\text{target,x}} - 0.2, p^{\text{target,y}}, p^{\text{target,z}}\} \forall i \in \{111, 112, ..., 146\}$$

B. Trajectory filtering

As DreamControl does not consider scene elements to avoid collision with like the platform on which the target object is kept, some of the generated trajectories turn out to be infeasible to be used for reference trajectory in stage 2. Also, some of the generated trajectories empirically turned out to be very dynamically infeasible in practice to be tracked by RL in stage 2. Hence, we filter out trajectories based on some heuristics like reject a trajectory if it collides with the scene environment, reject if it bends it's waist too much etc.

TABLE XI: # of filtered trajectories and constants used for filtering

Task	#Trajs before	#Trajs after	β_{torso}	β_{pelvis}
`Pick`	100	67	$\pi/4$	0.6
`Precise Punch`	100	100	$\pi/4$	0.6
`Precise Kick`	100	66	$\pi/2$	0.5
`Press Button`	100	96	$\pi/3$	0.5

in the trajectory. Specifically, these are the conditions used to filter out trajectories:-

- Reject if torso angle with x axis is larger than β^{torso} i.e. reject if $\arccos(axis_x^{\text{torso,x}}) > \beta^{\text{torso}}$ where $axis_z^{\text{torso}}$ is the x axis of the torso. This is to reject reference trajectories where the robot bends too much or turns around which is undesired for tasks in question
- Reject if pelvis height is below a certain threshold β^{pelvis}. This is to reject reference trajectories where robot squats too much for tasks it is not needed to.
- Reject if any part of the body collides with the scene

The specific thresholds, β^{torso} and β^{pelvis} for tasks that we do filtering and number of filtered trajectories from total sampled are given in Table XI. For all other tasks except `Open Drawer` and `Open Door`, only 1 trajectory is manually selected for training which can be easily scaled up with careful engineering for their automated filtering. For `Open Drawer` and door tasks, as trajectories are obtained by optimization of right arm, it does not need filtering. It must be noted that these heuristic-based filtering is only required because diffusion-generated trajectories are unfit for tracking. This maybe mainly because diffusion model sees out-of-distribution samples during annealing. With more data, the need for filtering out samples maybe eliminated.

C. Trajectory refinement

The trajectories obtained after re-targeting and filtering do not start from a same pose. This is a characteristic of DreamControl that it only restricts the $p^{\text{ref root, x}}$, $p^{\text{ref root, y}}$ and θ^{yaw} to start from origin at $t = 0$ but the generated motion could start from any pose at $t = 0$. However, for RL training in stage 2, we need reference trajectories to start from a fixed pose and point. Hence, in each generated reference trajectory, we prepend a trajectory that starts from a fixed default joint pose at fixed pose of the root and interpolates to the start joint pose and root pose of the generated trajectory. Specifically, we append $N^{\text{init}} = 20$ frames at the beginning of each motion of which for the first 10 frames the trajectory remains static and next 10 frames to interpolate to α_0, to obtain refined reference trajectory, α^{refined} consisting of 216 frames.

Further, we refine motions by disabling movement of the non-functional arm based on the task to avoid it's unnecessary movement in the reference trajectory. Specifically, we set the joint angles of the left arm and right arm group joints denoted as $G^{\text{left arm}}$ and $G^{\text{right arm}}$ as in Table VII to their default values along all time steps. The default joint angles. The default values are given in Table VI. The specific group that was refined to set to default joint angles for each task is given in Table VII. The tasks not included in table do not have this refinement applied.

1) Special case for `Pick` task: Last, specifically for `Pick` task, we observed that as our robot is shorter ($1.32m$) than the SMPL model used to train generative model on ($1.74m$), most motions in the dataset are of the human picking object from platform which is located near or below it's waist while for the shorter robot, the platform is more close to it's shoulder above it's waist. Hence, most generated motion when the target point is higher for go through the platform. Hence, specifically for the `Pick` task, we add an additional refinement layer before passing them for filtering to minimally modify the reference trajectory to avoid collision of right hand with the platform. Specifically, we solve the following optimization problem through gradient descent with joint angle variables initialized from the reference trajectory:-

$$\begin{aligned} q^{\text{ref},*} = argmin\Sigma_{t=\{\Delta t,\ldots,(L-1)\Delta t\}} | (\| p_t^{\text{right hand},*} - p_{t-1}^{\text{right hand},*} \|_2 \\ - \| p_t^{\text{right hand}} - p_{t-1}^{\text{right hand}} \|_2) | \\ s.t. \;\; d(p_t^{\text{right hand}}) = 0 \end{aligned}$$

where $d(.)$ is a function that maps 3D points to their closest distance from the free space. The rationale for choosing relative distance difference from reference as the objective is to preserve the smoothness of the motion in the generated trajectory but just modify the trajectory minimally to avoid collision of right hand with the platform. That is if the reference trajectory was to slow down while approaching the object as humans do, that slowing down will still be preserved in the trajectory after modifying it to avoid collision. If after making this refinement, if some other body part like feet in the reference trajectory makes collision with the platform, that trajectory is rejected.

D. Sim2Real

For sim2real trajectory generation and refinement, we follow a similar design to their corresponding sim tasks with some changes in prompt and trajectory refinements that we will discuss in this section. There are 5 tasks that we put on hardware robot. Specific prompts used for them are listed in Table X. The generated trajectories are also slowed down by a certain factor as also reported in Table X. This is to ensure safety of the motion and executed and minimize the sim-real gap by making the motion slower. It was empirically observed that with the same speed, due to some sim-to-real gap, the executed trajectories exhibited less success rates for precise tasks like pick where the object got pushed a little bit and thus failed to be grasped as the position is not updated in the current design. This slowing down was not required for `Squat` and `Bimanual Pick` tasks. In addition to the standard refinements as made in the sim versions of these tasks (except `Squat`), the specific spatial control points and refinements for each task are as follows:-

1) `Pick:` For each reference trajectory:- We sample a target point, $p^{\text{target}} = \{p^x \in \mathcal{U}(0.25, 0.35), p^y \in \mathcal{U}(-0.3, 0.0), p^z = 1.1\}$. Target time-step is chosen to be, $t'_g = 30$. Then, we set spatial control signal for wrist as follows:-

$$\lambda_i^{\text{right\_wrist}} = p^{\text{target}} \forall i \in \{t'_g, ..., t'_g + 20\}$$
$$\lambda_i^{\text{right\_wrist}} = (p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}} + 0.2)$$
$$\forall i \in \{t'_g + 20, ..., t'_g + 40\}$$

We also set the target spatial points for elbow to encourage generating trajectories where the object is grabbed from side as follows:-

$$\lambda_i^{\text{elbow}} = (p^{\text{target,x}}, p^{\text{target,y}} - 0.26\cos(\frac{\pi}{4}), p^{\text{target,z}} - 0.26\sin(\frac{\pi}{4}))$$
$$\forall i \in \{t'_g, ..., t'_g + 20\}$$

Of the generated trajectories, we refine them to avoid collision with the platform, plus we also add an additional cost to bring the trajectories close to their corresponding goal points for which the trajectories were generated.

$$q^{\text{ref},*} = argmin \Sigma_{t=\{\Delta t, ..., (L-1)\Delta t\}} |(\|p_t^{\text{right hand},*} - p_{t-1}^{\text{right hand},*}\|_2 - \|p_t^{\text{right hand}} - p_{t-1}^{\text{right hand}}\|_2)| + M_t \|p_t^{\text{right wrist}} - \lambda_t^{\text{right wrist,'}}\|$$
$$s.t. \;\; d(p_t^{\text{right hand}}) = 0$$

Where $M_t = 1$ for $t \in \{0, t'_g, t'_g + 1, ..., t'_g + 40\}$ and 0 otherwise. This is to mask for timesteps where the goal is specified. Idea is to minimally modify the trajectories such that they follow the corresponding goal. This refinement was required to ensure smooth policies. We also set the lower body (groups $G^{\text{left leg}}$ and $G^{\text{right leg}}$) to fixed default joint angles and adjust the root height such that the feet touches the ground. This is to enforce that the motion is stand still.

2) `Precise Punch:` For each reference trajectory:- We sample a target point, $p^{\text{target}} = \{p^x \in \mathcal{U}(0.6, 0.65), p^y \in \mathcal{U}(-0.2, 0.1), p^z = \mathcal{U}(1.15, 1.4)\}$. Target time-step is chosen to be, $t'_g = 30$. Then, we set spatial control signal for wrist as follows:-

$$\lambda_i^{\text{right\_wrist}} = p^{\text{target}} \forall i \in \{t'_g, ..., t'_g + 20\}$$
$$\lambda_i^{\text{right\_wrist}} = (p^{\text{target,x}}, p^{\text{target,y}}, p^{\text{target,z}} + 0.2)$$
$$\forall i \in \{t'_g + 20, ..., t'_g + 40\}$$

Of the generated trajectories, we refine them to bring the trajectories close to their corresponding goal points for which the trajectories were generated.

$$q^{\text{ref},*} = argmin \Sigma_{t=\{\Delta t, ..., (L-1)\Delta t\}} |(\|p_t^{\text{right hand},*} - p_{t-1}^{\text{right hand},*}\|_2 - \|p_t^{\text{right hand}} - p_{t-1}^{\text{right hand}}\|_2)| + M_t \|p_t^{\text{right wrist}} - \lambda_t^{\text{right wrist,'}}\|$$
$$s.t. \;\; d(p_t^{\text{right hand}}) = 0$$

Where $M_t = 1$ for $t \in \{0, t'_g\}$ and 0 otherwise. This is to mask for timesteps where the goal is specified. Idea is to minimally modify the trajectories such that they follow the corresponding goal. This refinement was required to ensure smooth policies. We also set the lower body (groups $G^{\text{left leg}}$ and $G^{\text{right leg}}$) to fixed default joint angles and adjust the root height such that the feet touches the ground. This is to enforce that the motion is stand still.

3) `Bimanual Pick:` We follow the exact same spatial control points as the sim to generate the trajectory. We additionally refine to dismiss any feet slipping by using IK to adjust $G^{\text{left leg}}$ and $G^{\text{right foot}}$ to bring left and right foot to their positions at $p_0^{\text{left foot}}$ and $p_0^{\text{right foot}}$. Other than this, we set the $roll$ and yaw of root to 0 for symmetry.

4) `Open Drawer:` We follow the exact same procedure as the open drawer trajectories for sim.

5) `Squat:` For each trajectory, we sample a trajectory with target point, $p^{\text{target}} = \{p^x = -0.15, p^y = 0.0, p^z = \mathcal{U}(0.4, 0.6)\}$. Target time-step is chosen to be $t'_g = 100$. We set the spatial control signal for pelvis as: $\lambda_i^{\text{pelvis}} = p^{\text{target}}$ for $i = t'_g$. We additionally refine to dismiss any feet slipping by using IK to adjust $G^{\text{left leg}}$ and $G^{\text{right foot}}$ to bring left and right foot to their positions at $p_0^{\text{left foot}}$ and $p_0^{\text{right foot}}$. Other than this, we set the $roll$ and yaw of root to 0 for symmetry.

X. RL Training

A. Model architecture

We use a simple fully-connected MLP to represent both our policy (or actor) and critic for each task. The network architecture for the actor and critic has the following hidden layers: $(512, 256, 256)$. Also, we use the same observations for policy and critic as opposed to asymmetric actor-critic setup in [73].

B. Environment parameters

We randomize the object/target location (by sampling different trajectories in some tasks), object mass, friction of surface/object. All of these variations for each task are listed in Table XII.

C. Task-specific sparse rewards

We add task-specific sparse rewards that are chosen to be indicative of whether the task is successful as well. These rewards are listed in Table XIII. For each task, we have a body part link, b that maybe used in the definition of the sparse rewards and the time t_g when the interaction is supposed to happen.

TABLE XII: Environment randomization parameters for each task

Task	Friction	Mass of object
`Pick`	$\mathcal{U}(0.7, 1)$	$\mathcal{U}(0.1, 1)$
`Precise Punch`	$\mathcal{U}(0.7, 1)$	-
`Precise Kick`	$\mathcal{U}(0.7, 1)$	-
`Press Button`	$\mathcal{U}(0.7, 1)$	-
`Jump`	$\mathcal{U}(0.7, 1)$	-
`Sit`	$\mathcal{U}(0.7, 1)$	-
`Bimanual Pick`	$\mathcal{U}(0.7, 1)$	$\mathcal{U}(0.1, 5)$
`Pick from Ground (Side Grasp)`	$\mathcal{U}(0.7, 1)$	$\mathcal{U}(0.1, 1)$
`Pick from Ground (Top Grasp)`	$\mathcal{U}(0.7, 1)$	$\mathcal{U}(0.1, 0.5)$
`Pick and Place`	$\mathcal{U}(0.7, 1)$	$\mathcal{U}(0.1, 0.5)$
`Open Drawer`	$\mathcal{U}(0.7, 1)$	-
`Open Door`	$\mathcal{U}(0.7, 1)$	-

TABLE XIII: Task-specific sparse rewards for each task

Task	Body part link, b	Sparse reward, r_t^{sparse}	Description
`Pick`	right_wrist_yaw_link	$1_{h_t^{\text{object}} > h^{\text{thres}}} 1_{t \geq t_b}$	$h^{\text{thres}} = 0.95$, h_t^{object} is height of object
`Precise Punch`	right_wrist_yaw_link	$1_{\|p_t^b - p^{\text{goal}}\|_2 < d^{\text{thres}}} 1_{t \geq t_b - 0.1} 1_{t \leq t_b + 0.1}$	$d^{\text{thres}} = 0.05$
`Precise Kick`	right_ankle_roll_link	$1_{\|p_t^b - p^{\text{goal}}\|_2 < d^{\text{thres}}} 1_{t \geq t_b - 0.1} 1_{t \leq t_b + 0.1}$	$d^{\text{thres}} = 0.1$
`Press Button`	right_wrist_yaw_link	$1_{\|p_t^b - p^{\text{goal}}\|_2 < d^{\text{thres}}} 1_{t \geq t_b - 0.1} 1_{t \leq t_b + 0.1}$	$d^{\text{thres}} = 0.05$
`Jump`	pelvis	$1_{\|p_t^b - p^{\text{goal}}\|_2 < d^{\text{thres}}} 1_{t \geq t_b - 0.1} 1_{t \leq t_b + 0.1}$	$d^{\text{thres}} = 0.1$
`Sit`	pelvis	$1_{\|p_t^b - p^{\text{goal}}\|_2 < d^{\text{thres}}} 1_{t \geq t_b}$	$d^{\text{thres}} = 0.05$
`Bimanual Pick`	right_wrist_yaw_link, left_wrist_yaw_link	$1_{h_t^{\text{object}} > h^{\text{thres}}} 1_{t \geq t_b}$	$h^{\text{thres}} = 0.7$, h_t^{object} is height of object
`Pick from Ground (Side Grasp)`	left_wrist_yaw_link	$1_{h_t^{\text{object}} > h^{\text{thres}}} 1_{t \geq t_b}$	$h^{\text{thres}} = 0.2$, h_t^{object} is height of object
`Pick from Ground (Top Grasp)`	right_wrist_yaw_link	$1_{h_t^{\text{object}} > h^{\text{thres}}} 1_{t \geq t_b}$	$h^{\text{thres}} = 0.3$, h_t^{object} is height of object
`Pick and Place`	right_wrist_yaw_link	$1_{\|p_t^{\text{object}} - p^{\text{goal}}\|_2 < d^{\text{thres}}} 1_{t \geq t_b}$	$d^{\text{thres}} = 0.1$, p^{goal} is position of the goal
`Open Drawer`	right_wrist_yaw_link	$1_{a_t^{\text{drawer}} > a^{\text{thres}}} 1_{t \geq t_b}$	a_t^{drawer} is drawer open amount, $a^{\text{thres}} = 0.05$
`Open Door`	right_wrist_yaw_link	$1_{a_t^{\text{door}} > a^{\text{thres}}} 1_{t \geq t_b}$	a_t^{door} is door open amount, $a^{\text{thres}} = 0.05$

D. Task-specific dense rewards for TaskOnly+

We also add a task-specific dense reward to encourage pre-grasp/pre-approach pose for the object/goal respectively. This task-specific reward is adapted from [8] and is defined as follows:-

$r_t^{\text{dense}} = \|p_{t-\Delta t}^{\text{b}} - p_{t_g}^{\text{ref,b}}\|_2 - \|p_t^{\text{b}} - p_{t_g}^{\text{ref,b}}\|_2$.

The specific body parts used in each task are listed in Table XIII.

E. Reward weights for each task

The rewards weights for each task are listed in Table XIV.

TABLE XIV: Reward weights for all tasks

Task	**Tracking**					**Smoothness**					$w_{r,\text{task,sparse}}$	$w_{r,\text{task,dense}}$
	w_{r_1}	w_{r_2}	w_{r_3}	w_{r_4}	w_{r_5}	w_{r_6}	w_{r_7}	w_{r_8}	w_{r_9}	$w_{r_{10}}$		
`Pick`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.5	-1	0.1	100
`Precise Punch`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.5	-1	1	100
`Precise Kick`	-0.2	-0.1	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.15	-1 for left, -0.3 for right	1	100
`Press Button`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.5	-1	1	100
`Jump`	-0.2	-0.1	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	0	-1	1	100
`Sit`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.5	-1	1	100
`Bimanual Pick`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	0	-1	0.1	100
`Pick from Ground (Side Grasp)`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.5	-1	0.1	100
`Pick from Ground (Top Grasp)`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.15	-1	0.1	100
`Pick and Place`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	-0.5	-1	0.1	100
`Open Drawer`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	0	-1	0.1	100
`Open Door`	-0.2	-0.05	-0.2	-0.2	0.3	-1.5e^{-7}	-5e^{-3}	-0.1	0	-1	0.1	100