LATENT JACOBIAN STEER: ELICITING CONTACT-RICH OUT-OF-DISTRIBUTION BEHAVIORS FROM PRETRAINED HUMANOID CONTROLLERS

S K Swaminathan[1,2,*] **Shivam Sood**[1] **Shreyansh Kansal**[1]
Chengyang He[1] **Guillaume Adrien Sartoretti**[1]

[1]National University of Singapore [2]Indian Institute of Technology Kharagpur
https://latent-jacobian-steer.github.io *swami@u.nus.edu

ABSTRACT

Given a pretrained general latent-conditioned whole-body humanoid controller, say a behavioral foundation model (BFM), how can we elicit complex and contact-rich out-of-distribution behaviors? Raw joint-action exploration enables direct, task-specific adjustments but can disrupt coordination learned by the pretrained controller, whereas latent exploration retains that coordination but is difficult to steer toward desired physical effects. To combine the complementary strengths of these exploration spaces, we propose Latent Jacobian Steer (LJS), which couples sampling in direct joint-action space with the semantics of the learned frozen controller's input latent space. For a task specified through desired physical effects, we sample task-relevant joint-action perturbations and map them through a damped local inverse of the policy Jacobian to obtain candidate latent (z) commands. These commands are executed through the frozen pretrained policy, while exploration is directed toward the physical outcomes required by the new task. Candidate outcomes are evaluated in simulation and composed into latent-command sequences that accomplish the task. Ablations show that neither action-space nor latent-space exploration *alone* provides the same capability as their coupling. Across diverse out-of-distribution tasks, simulation-based searching using this method produces contact-rich behaviors, including hurdling, manipulating articulated objects, and opening doors through non-prehensile contact. Over ten seeds, LJS completes more of each humanoid task than when searching in either space alone. Beyond humanoids, it also steers a goal-conditioned dexterous hand policy through four assembly tasks better than searching in goal space directly.

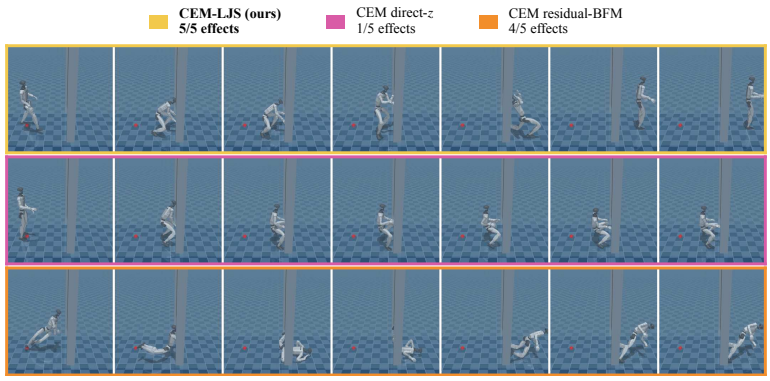


Figure 1: **Shutter traversal** Chronological frames from comparing CEM-based (Rubinstein, 1999) searches of our method and the best seeds of baseline methods over ten seeds. **CEM-LJS(ours)** lifts the shutter, crosses on its feet, and stands up. CEM with direct-z search never lifts the shutter. CEM with residual raw joint-action search over BFM hacks the task by crawling on its torso.

1 INTRODUCTION

For humanoid robots to become useful, they must be able to physically interact with objects and environments in diverse and often contact-rich ways. Recent advances in motion capture and general motion-tracking have enabled humanoids to reproduce an increasingly broad repertoire of dynamic whole-body behaviors (Chen et al., 2025; Liao et al., 2026). Yet obtaining behaviors for a new contact-rich loco-manipulation task remains considerably more task-specific. Such behaviors often require dedicated human demonstrations or trajectories reconstructed from video through Real2Sim pipelines that recover the relevant object and environment interactions (Dan et al., 2025; Ranawaka et al., 2026).

High-fidelity simulation provides a natural way to generate such behaviors. Existing approaches using simulation either learn new behaviors from task-specific rewards (Sferrazza et al., 2024; Mu et al., 2024) or explicitly generate demonstrations and trajectories for them (Jing et al., 2025; Taouil et al., 2026). More recently, Behavioral Foundation Models such as Meta Motivo and BFM-Zero have instead pretrained general latent-conditioned whole-body controllers that can be prompted towards different motions, goals, and rewards (Tirinzoni et al., 2025; Li et al., 2026b). These approaches inspire our question: can we steer such a frozen controller to produce complex contact-rich behaviors that lie outside its training distribution, without collecting a task-specific motion demonstration or training a new policy?

We hypothesize that much of the whole-body coordination required for these behaviors is already present in a sufficiently general pretrained controller, even if it was never trained on such object-environment interaction data. This is fairly intuitive in hindsight. A human who has learned to move in free space can use these motion priors to play around with objects and the environment. The challenge is how to *steer* it towards a new physical interaction. Direct exploration in joint-action space makes task-specific changes easy to express, for example, moving a hand towards a desired contact, but it can disrupt the coordination learned by the controller. Exploration in its latent space preserves this coordination, but it is difficult to know which latent perturbation will produce a desired physical effect. Our key observation is that these two spaces are locally connected by the frozen policy itself: its Jacobian describes how perturbations in latent commands change the resulting joint actions.

Based on this observation, we introduce **Latent Jacobian Steer (LJS)** (Figure 2), which couples precise exploration in joint-action space with coherent execution through the latent-conditioned pretrained controller. Given a task-relevant joint-action perturbation, LJS uses a damped local inverse of the policy Jacobian to obtain candidate latent commands, which are then executed and evaluated in simulation. Complex tasks can then be described through simple sequences of physical effects: opening a door, for example, can be specified as (1) reaching and grasping the handle, (2) manipulating it to release the latch, and (3) pushing the door about its hinge. In this way, LJS provides a means of eliciting and collecting new contact-rich behaviors from a frozen whole-body controller without requiring a task-specific motion demonstration, a complete reference trajectory, or training a new task policy. We evaluate LJS over ten seeds on three contact-rich humanoid tasks and, beyond humanoids, on four precise assembly tasks with a frozen goal-conditioned dexterous hand policy.

2 RELATED WORKS

From motion tracking to contact-rich behavior discovery. General trackers such as TWIST (Ze et al., 2025) and SONIC (Luo et al., 2026) provide broad whole-body motion repertoires, while PGMT (Li et al., 2026a) and GigaBrain-WBC-0.5 (Cheng et al., 2026) incorporate terrain perception and environment interaction. Motion generators such as OMOMO (Li et al., 2023) and CHOIS (Li et al., 2024) produce human–object interactions from object trajectories or language and spatial guidance, using captured interactions as training data. For a new contact-rich task, however, the movement that realizes the required interaction must still be obtained. Real2Sim pipelines reconstruct it from demonstrations (Dan et al., 2025; Ranawaka et al., 2026); contact-implicit trajectory optimization (Posa et al., 2014) and logic-geometric programming (Toussaint, 2015) optimize motions and manipulation decisions. More recently, MotionDisco (Taouil et al., 2026) combines motion search with trajectory optimization, while OmniContact (Yu et al., 2026b) chains skills through contact-flow representations. We investigate whether these interactions can instead be discovered by

steering the existing coordination of a frozen whole-body controller, without supplying a reference motion or training a new task policy.

Reusing and steering pretrained behaviors. Latent skill methods such as SPiRL (Pertsch et al., 2020), OPAL (Ajay et al., 2021), SkiMo (Shi et al., 2023), and ASE (Peng et al., 2022) provide reusable primitives for downstream learning or planning. METRA (Park et al., 2024b) and HILP (Park et al., 2024a) learn behavioral spaces before downstream tasks are specified, while forward-backward representations (Touati & Ollivier, 2021) support reward-based policy inference without retraining. This direction underlies whole-body behavioral models including Meta Motivo (Tirinzoni et al., 2025), BFM-Zero (Li et al., 2026b), and FB-MEBE (Hu et al., 2026). Beyond whole-body control, Play2Perfect (Lum et al., 2026) pretrains a goal-conditioned dexterous hand policy through play and then fine-tunes it for each precise assembly task. Their behavioral coverage provides a useful starting point, but finding commands for precise new interactions remains difficult. Adaptation methods address this through parameter updates, skill-based learning, or action corrections, as in OpenVLA (Kim et al., 2025), SUPE (Wilcoxson et al., 2025), and residual RL (Johannink et al., 2019). Closer to our setting, FlowDAgger (Murray et al., 2026), UniSteer (Lu et al., 2026), and Flow Reversal Steering (Tang et al., 2026) map human or VLM corrections into the noise space of frozen generative policies. These works connect action-space guidance to latent execution. LJS instead discovers useful corrections from simulator-evaluated physical outcomes, without expert corrective actions or a learned steering policy.

Searching through the policy's local inverse. CEM (Rubinstein, 1999), iCEM (Pinneri et al., 2020), MPPI (Williams et al., 2017), and DIAL-MPC (Xue et al., 2025) search through simulated outcomes; SkiMo (Shi et al., 2023) and BFM-Zero (Li et al., 2026b) also demonstrate planning over learned behavioral spaces. Our question concerns how to construct useful proposals within a frozen controller. Structured exploration methods such as gSDE (Raffin et al., 2022), LATTICE (Chiappa et al., 2023), and DEP-RL (Schumacher et al., 2023) organize perturbations for coordinated control, while Safe Mutations (Lehman et al., 2018) and Policy Manifold Search (Rakicevic et al., 2021) use output sensitivities to guide exploration. Drawing on local inverse mappings used in damped inverse kinematics (Wampler, 1986) and operational-space control (Khatib, 1987), LJS couples joint-action proposals to latent execution through the policy Jacobian. CEM searches joint-command changes, a damped inverse maps them into latent perturbations, and simulator rollouts evaluate their physical effects. The Jacobian describes the frozen actor's local response, not contact dynamics. This coupling lets joint-action coordinates structure the search while the pretrained controller generates every executed command.

3 Latent Jacobian Steer

We propose Latent Jacobian Steer (LJS), which couples the direct joint-action and latent search spaces. We search for task-relevant changes in the joint-action space, map them to changes in the latent command via the local Jacobian of the frozen policy, and evaluate the resulting behavior using physics simulation. Figure 2 gives an overview.

Problem Formulation. We consider humanoid control in a simulator as a finite-horizon Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{U}, P, \rho_0, T)$, where $\mathcal{S}$ is the state space, $\mathcal{U}$ is the joint-command space, P is the transition dynamics including the low-level controller, ρ_0 is the initial-state distribution, and T is the task horizon. We specify the task objective separately via a *physical-effect* sequence $\mathcal{T}$, defined below, whose scores guide the search and whose acceptance conditions determine task completion. The state s_t includes the robot and environment configurations and velocities, together with the controller memory needed to make the transitions Markovian. For the 29-DOF humanoid considered here, $u_t \in \mathcal{U} \subseteq \mathbb{R}^{29}$ specifies joint-position targets, realized through a joint-space PD controller. We have a frozen latent-conditioned pretrained policy $u_t = \pi_\theta(h_t, z_t)$, where h_t contains the robot's proprioceptive observations and action history, and $z_t \in \mathcal{Z}$ is its latent command. For our particular purpose of demonstrating the adaptation of the pretrained policy over interesting out-of-distribution (OOD) tasks, we begin by concretizing *how to define a task itself*. It is worth noting that we chose this as our task definition for ease of expressing various tasks, and we are not proposing it as a canonical way to define loco-manipulation tasks. We define a task as an ordered sequence of physical effects $\mathcal{T} = (E_1, \ldots, E_K)$. Each effect E_k specifies a physical measurement ϕ_k, a desired target or target region Y_k, and a non-negative loss ℓ_k. For a trajectory segment ξ, its physical-effect score

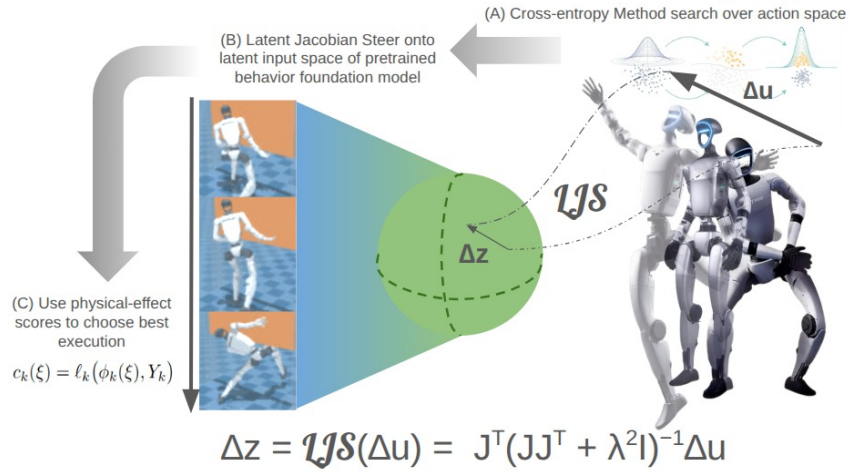


Figure 2: **Latent Jacobian Steer overview.** (A) CEM proposes joint-action perturbations Δu. (B) A damped inverse of the frozen policy Jacobian maps each proposal to a latent change Δz; the updated latent command is normalized and simulated through the pretrained controller. (C) Physical-effect scores rank the rollouts and guide CEM updates. The best admissible candidate is executed for a short window before replanning. The actor remains frozen throughout.

is $c_k(\xi) = \ell_k\big(\phi_k(\xi), Y_k\big)$. The measurement may describe body or object displacement, orientation, contact forces, or conditions that must hold over the segment. When an effect requires several conditions simultaneously, we combine their losses: $c_k(\xi) = \sum_{j=1}^{m_k} w_{k,j}\ell_{k,j}\big(\phi_{k,j}(\xi), Y_{k,j}\big), w_{k,j} > 0$. We also specify an acceptance condition $A_k(\xi)$ for each effect. By default, an effect is accepted when all its component losses are zero; separate acceptance tolerances can be specified where needed. This distinguishes the score used to guide exploration from the condition used to decide whether the physical effect has been achieved. The task is satisfied when the executed trajectory admits ordered boundaries $0 = t_0 \le t_1 \le \cdots \le t_K \le T$ such that $A_k\big(\xi_{t_{k-1}:t_k}\big) = 1; k = 1, \ldots, K$. Any condition that must persist across multiple effects is included in each relevant effect. The subsequent search has access to privileged simulator state and contact information to evaluate the desired physical effects; the pretrained policy continues to receive its original observations. Our goal is to find a sequence of latent commands that satisfies these effects without changing θ, that is, fine-tuning or retraining the policy. Here, OOD tasks require contact interactions and physical effects that are not represented during the controller's pretraining.

The choice of pretrained controller is important for LJS. Any latent-conditioned policy or general motion tracker is not necessarily suitable simply because it supports diverse behaviors: its latent space must also be meaningfully steerable toward downstream objectives. For our main experiments, we use BFM-Zero (Li et al., 2026b), which learns a shared latent space through unsupervised reinforcement learning with forward-backward representations and can be prompted using motions, goals, or rewards without task-specific retraining. This is particularly useful in our setting, where we specify the desired physical effect without providing a reference motion that realizes it. We therefore use LJS to explore how to steer this latent space toward new contact-rich behaviors.

3.1 Our choice of latent-conditioned pretrained controller, BFM-Zero

BFM-Zero (Li et al., 2026b) learns a family of whole-body behaviors through a latent-conditioned policy $\pi_\theta(h, z)$. Its forward-backward (FB) representation relates latent commands to the future states the policy visits, allowing motions, goals, and rewards to prompt the same controller without task-specific retraining. We note that BFM-zero has *not* been trained on environment interaction or loco-manipulation behaviors.

For the Unitree-G1-specific policy (Li et al., 2026b) used in our work, the latent command lies on the sphere $\mathcal{Z} = \{z \in \mathbb{R}^{256} : \|z\|_2 = 16\}$. The actor produces 29 joint-position targets at 50 Hz.

LJS searches these latent commands while keeping the actor frozen. We defer the FB formulation and its prompting mechanisms to Appendix A.1.

3.2 Defining the physical effects for a contact-rich OOD task

As described earlier, we define a task through a small sequence of measurable physical outcomes. To illustrate: for shutter traversal, this involves placing the palms beneath the shutter, lifting it, moving the head and torso through the opening, clearing it with the entire body, and standing upright on the far side. Each effect specifies what must happen while leaving the movements that realize it to the search. For example, lifting can require a minimum shutter height and sufficient upward support, with a score penalizing shortfalls and a separate acceptance condition determining completion. It is to be noted that defining these effects are not equivalent to dense reward engineering, as the underlying motion is not engineered and depends heavily on the frozen controller.

We construct these effects stage-wise, using an LLM as a coding assistant to translate intuitive task descriptions into simulator measurements and scores. We inspect rollouts and refine definitions that admit unintended outcomes. We defer the mathematical score definitions, task-specific sequences, and acceptance conditions to Appendix K.

3.3 Coupling of the latent and direct action space

Once the physical effect for a particular stage of a task is defined, our goal is to find latent commands that satisfy it. We do this by searching in joint-action space using CEM (Rubinstein, 1999) and pulling the proposed changes back into the latent space through the frozen policy's local Jacobian.

How is a joint-action change pulled back into the latent space? Let $u(h, z)$ denote the joint-position targets produced by the frozen policy, including its output scaling and latent normalization. At the current observation history h and latent command z, we compute

$$J(h, z) = \frac{\partial u(h, z)}{\partial z} \in \mathbb{R}^{29\times 256}.$$

This is the policy Jacobian: it describes how the current joint targets change with the latent input while h is held fixed. For a small latent change,

$$u(h, z + \Delta z) - u(h, z) \approx J(h, z)\Delta z.$$

Thus, given a proposed joint-action change Δu, we seek a latent change that approximately realizes it. Since J is rectangular and may have weakly responsive directions, we use a damped least-squares inverse:

$$\Delta z^\star = \arg\min_{\Delta z} \|J\Delta z - \Delta u\|_2^2 + \lambda^2\|\Delta z\|_2^2, \qquad \lambda > 0.$$

Setting the gradient to zero gives

$$(J^\top J + \lambda^2 I)\Delta z^\star = J^\top \Delta u,$$

and hence

$$\Delta z^\star = \underbrace{J^\top (JJ^\top + \lambda^2 I)^{-1}}_{A} \Delta u.$$

The damping limits latent changes along directions in which the policy is insensitive. We then form the candidate

$$z' = \Pi(z + A\Delta u), \qquad \Pi(v) = 16\frac{v}{\|v\|_2}.$$

The Jacobian includes differentiation through Π, so the local change accounts for the controller's spherical latent domain. This mapping approximates the policy's immediate output response.

How do we search for joint-action changes? We adapt the CEM implementation in Hydrax (Kurtz, 2024) to search over joint-action perturbations. At each planning step, we initialize a zero-mean diagonal Gaussian and divide the candidate budget across several CEM rounds. Each sampled perturbation is mapped into a latent command: $\Delta u_i^{(q)} \sim \mathcal{N}\left(\mu_q, \mathrm{diag}(s_q^2)\right), z_i^{(q)} = \Pi\left(z + A\Delta u_i^{(q)}\right)$. We simulate the candidates, rank them by their physical-effect scores, and refit the Gaussian to the

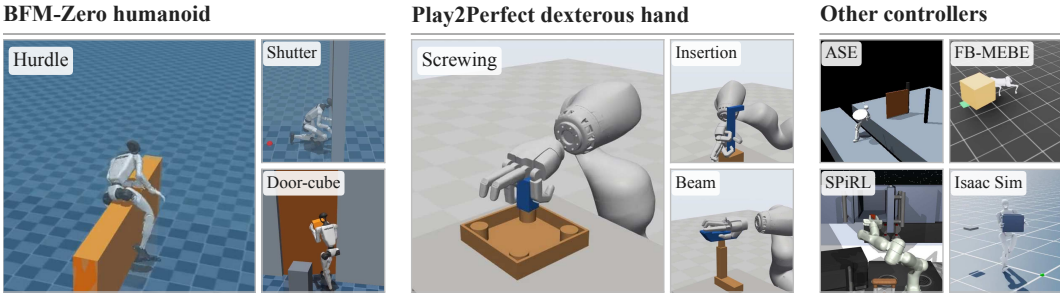


Figure 3: **Behaviors elicited from frozen pretrained controllers with LJS.** Left: the BFM-Zero humanoid controller. Middle: the Play2Perfect dexterous hand policy. Right: further controllers and a second simulator (Appendix I). No controller was trained or fine-tuned for these tasks.

best admissible joint-action proposals. Each round includes its current mean, so the first round also tests the unchanged command. A minimum standard deviation retains exploration; if no candidate is admissible, the distribution is left unchanged. We defer the full CEM updates and elite-selection rules to Appendix A.2.

How do we score and execute a candidate? We simulate each candidate for H control steps from the same saved simulator state and controller history, holding its latent command fixed while the frozen actor receives updated observations. We rank candidates by their mean physical-effect score over the latter half of the rollout, allowing time for the response to develop. Candidates violating full-segment constraints are rejected and excluded from the elite set.

After all CEM rounds, we execute the lowest-scoring admissible candidate for $L \leq H$ steps. Its latent command becomes the new anchor; we recompute the Jacobian and restart CEM with zero-mean perturbations. We advance only when the executed segment satisfies the current effect's acceptance conditions. Search ends when all effects are accepted, the budget is exhausted, or no admissible candidate is found. The exact rollout objective and search parameters are provided in the appendix.

4 Experiments

We evaluate whether LJS can elicit contact-rich behaviors from a frozen whole-body controller, whether coupling joint-action proposals to latent execution improves task completion, and whether the same holds for a different pretrained controller. We run two sets of experiments: three contact-rich whole-body tasks with the frozen BFM-Zero G1 humanoid controller in MuJoCo (Todorov et al., 2012), and four precise assembly tasks with the frozen pretrained policy of Play2Perfect (Lum et al., 2026). Neither uses task-specific training or reference motions. Beyond task completion, we also examine search cost, motion quality, robustness to new starts and randomized physics, a sanity check with minimal effect definitions, and search ablations.

4.1 Contact-rich Tasks

Humanoid tasks with BFM-Zero. We consider three tasks that require different combinations of contact establishment, whole-body support, object manipulation, and recovery (Figure 3).

Hurdle traversal. The humanoid must cross a rigid hurdle using its body for support. It establishes palm and leg contacts, moves into a supported straddling configuration, transfers its body across the obstacle, places both feet on the far-side floor, and returns to standing. The ten effects require alternating support configurations, leg clearance, and recovery after crossing.

Shutter opening and traversal. The humanoid must place both palms beneath a vertically sliding shutter, lift it, move its head and torso through the opening, clear the panel with its entire body, and stand upright on the far side. These five effects combine precise palm placement and orientation with upward support, crouched whole-body movement, and recovery.

Door opening while carrying a cube. The humanoid approaches a cube on a stand, grasps its opposing side faces, lifts it clear, and carries it to a closed door. It then uses the grasped cube to depress

the handle and push the door open. The six effects require retaining the cube while manipulating another object; the final effect requires a door angle of at least $0.95\,\mathrm{rad}$, bilateral palm contact with the cube, and contact between the cube and door.

Assembly tasks with Play2Perfect. Our second set of experiments takes LJS beyond humanoids, to the pretrained play policy of Play2Perfect (Lum et al., 2026). This policy controls a Sharpa hand on a Franka arm through 29 joint targets and is conditioned on a 6-dimensional object goal. Play2Perfect fine-tunes the policy separately for each of its assembly tasks. We instead keep the pretrained policy frozen and only search over its goal input, on the same four tasks: placing the first and the second part of a beam assembly, screwing a leg onto its thread, and a tight peg insertion. Observe that the goal can be thought of loosely as a search variable that sequences and steers the policy over the whole episode, much like the latent command of BFM-Zero. Each task has three or four effects, for example picking up the leg, engaging the thread, turning it in, and releasing it for screwing. Here, LJS maps joint-action proposals through the Jacobian of the policy's joint targets with respect to its goal.

Other controllers and simulators. Since LJS only needs a frozen latent- or goal-conditioned policy and a simulator, we also show how it can be integrated with and used to steer the ASE (Peng et al., 2022) character controller to open a door and jump across a gap, the FB-MEBE (Hu et al., 2026) policy of a Go2 quadruped to push a box to a target, and the SPiRL (Pertsch et al., 2020) skill decoder of a robot arm to take a cube out of a cabinet. Finally, we also run BFM-Zero with LJS in a second simulator, Isaac Sim (Mittal et al., 2025), to carry a breadbox (Figure 3, Appendix I).

4.2 Defining the physical-effect scores

We construct effects stage-wise, beginning with an intuitive description of the required outcome. An LLM coding agent helps translate this description into simulator measurements, such as body-object distances, object orientation, contact forces, clearance, and velocity. These measurements define a score for ranking candidates and a separate acceptance condition for advancing to the next effect. They specify physical outcomes without prescribing the complete movement that realizes them.

We inspect the resulting rollouts on development seeds and revise definitions that admit unintended solutions, rerunning the current effect from the preceding accepted effect's terminal state. Every definition is human-reviewed and held fixed during evaluation; the LLM does not observe the search online or select commands. The evaluation seeds are never used while designing the effects. This back-and-forth required less than five hours to finalize a task's effects. All methods are evaluated with the same final task definitions. We note that the scores are geometric descriptions of physical outcomes and do not describe a behavior that only LJS can produce. The mathematical definitions and an example Human–AI interaction card (Feng et al., 2026) are provided in Appendix K; practical guidance is given in Appendix L.

4.3 Baselines and Evaluation

Our main comparisons keep CEM as the optimizer and change where proposals are made and how they are executed. We additionally ablate CEM while retaining the LJS coupling.

CEM-LJS. Our method searches over joint-action perturbations and maps each proposal into a latent command through the damped policy-Jacobian inverse. CEM fits its proposal distribution to the best simulated candidates, and the selected command is executed through the frozen actor.

CEM direct-z. We apply CEM directly to the 256-dimensional latent command, following the latent-optimization approach demonstrated by BFM-Zero (Li et al., 2026b). This retains the optimizer and frozen-policy execution while removing the joint-action proposal space and Jacobian pull-back. Note that, by itself, BFM-Zero's reward prompting cannot express our tasks, as its backward map sees only the robot's body, not the objects and contacts the effects measure (Appendix A.1).

CEM residual-BFM. CEM searches for additive corrections to the frozen actor's joint-position targets. This retains joint-action proposals but removes the requirement that they be realized through latent commands, giving the search direct authority to override the actor's output.

Mixed-Gaussian LJS. This optimizer ablation retains the Jacobian pull-back and replaces CEM with Gaussian proposals. Below a physical-effect score threshold, finite-difference simulator probes

estimate a targeted correction. The candidate population combines proposals around that correction and around zero, together with directional trials and the unchanged command.

CEM direct-goal. For Play2Perfect, CEM searches the 6-dimensional goal with the same budgets.

Evaluation. We evaluate every method on ten seeds that were never used while defining the tasks. For the humanoid tasks, all methods share one search setting: each candidate is simulated for 40 control steps, the selected command is executed for 20 steps, each decision evaluates 1,024 candidates (1,023 sampled and the unchanged command), and each effect allows at most 30 decisions. These settings are fixed across methods and seeds, to avoid any method-specific tuning. For Play2Perfect, candidates are simulated for 20 steps and executed for 10, with 1,024 candidates per decision (4,096 for the first beam part). We report the mean number of accepted effects, the number of fully completed tasks, and the wall-clock search time on an NVIDIA RTX 4090. Search cost counts simulated candidate control steps, including rejected candidates. For each effect k, we estimate the expected runtime (ERT) in these steps (Hansen et al., 2016):

$$\widehat{\mathrm{ERT}}_k = \frac{\sum_{i=1}^{10} C_{i,k}}{\sum_{i=1}^{10} \mathbf{1}\{\text{run } i \text{ reaches effect } k\}}.$$

Here, $C_{i,k}$ counts steps from task initialization to the first acceptance of effect k, or to termination if it is not reached. This estimates the cost of reaching the effect under independent restarts from the task's initial state, charging unsuccessful attempts as well.

4.4 Results

We ask the following questions to analyze whether LJS works as intended and what makes it robust.

Does coupling the two spaces complete more of each task? CEM-LJS completes the most effects on every humanoid task (Table 1), significantly so on the shutter and door-cube (Appendix C), and the gap grows with how precise the required contacts are. The shutter is the clearest example. Supporting and lifting the panel needs both hands placed beneath its edge and pushing upward, and CEM-LJS is the only method that gets through it. Direct latent search keeps the controller's coordination but cannot place the hands precisely enough to support the panel, so it almost never completes even the first effect. Residual search can move the hands to the right place, but overriding the joint targets also overrides the balance the controller provides. Its best run crawls under the panel with its torso on the floor and never stands back up (Figure 1). On door-cube, CEM-LJS completes the task on every seed, while residual search never gets the cube to the door. The hurdle is the hardest task for every method, and CEM-LJS gets the furthest on average.

Table 1: **Humanoid task completion with BFM-Zero, ten seeds.** Mean accepted effects ± s.d. (fully completed tasks).

Method	Hurdle (10 effects)	Shutter (5 effects)	Door-cube (6 effects)
CEM-LJS	**6.20 ± 3.43** (3/10)	**3.90 ± 2.08** (7/10)	**6.00 ± 0.00** (10/10)
CEM direct-z	4.60 ± 2.99 (1/10)	0.10 ± 0.32 (0/10)	4.90 ± 1.45 (6/10)
CEM residual-BFM	5.30 ± 3.92 (3/10)	1.50 ± 1.08 (0/10)	1.70 ± 1.25 (0/10)
Mixed-Gaussian LJS	3.60 ± 3.53 (1/10)	0.10 ± 0.32 (0/10)	4.10 ± 2.47 (5/10)

Does this carry over to another pretrained controller? The same pattern holds for a very different controller and embodiment. Steering the frozen Play2Perfect policy, CEM-LJS completes 31 of the 40 assembly runs, against 21 for direct goal search (Table 2). It completes at least as many runs as direct goal search on every task, and far more on screwing (Figure 4). Where the two differ, CEM-LJS also reaches the later effects at a much lower search cost (Figure 6). Searching the goal directly samples its six coordinates without regard to the hand, while LJS samples goal changes through their effect on the hand's joint targets, which matters most where the fingers must hold and turn the part.

How expensive is the search? Figures 5 and 6 show where the search spends its effort. The differences appear at the contact-critical effects, such as lifting the shutter or lifting and carrying the cube, where CEM-LJS needs far fewer candidate steps than the baselines. On the hurdle, residual

Table 2: **Assembly task completion with the frozen Play2Perfect policy, ten seeds.** Same format as Table 1.

Method	Beam, first part (3)	Beam, second part (4)	Screwing (4)	Insertion (3)
CEM-LJS	**3.00** ± **0.00** (10/10)	**3.30** ± **1.25** (7/10)	**3.80** ± **0.63** (9/10)	**2.00** ± **1.05** (5/10)
CEM direct-goal	**3.00** ± **0.00** (10/10)	2.30 ± 1.49 (4/10)	1.80 ± 1.62 (3/10)	1.80 ± 1.03 (4/10)

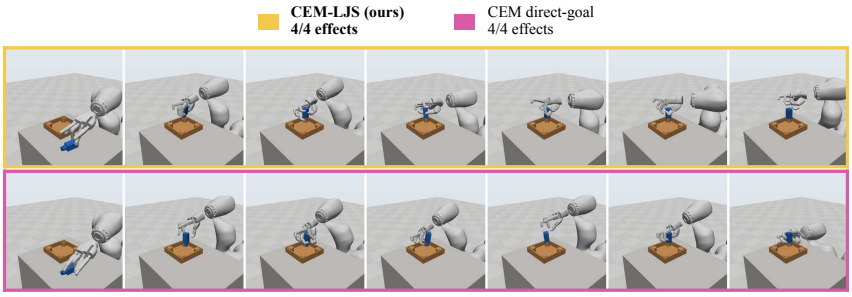


Figure 4: **Screwing with the frozen Play2Perfect policy.** Seven evenly spaced frames from a successful run of each method. CEM-LJS completes screwing from 9 of 10 seeds and CEM direct-goal from 3. Appendix H shows the other assembly tasks.

search reaches the later effects at a lower expected cost by correcting the joints directly, and this shows in its motion quality below. Most successful CEM-LJS searches finish within one to three minutes (Appendix C), which makes it practical to iterate on a task's effects.

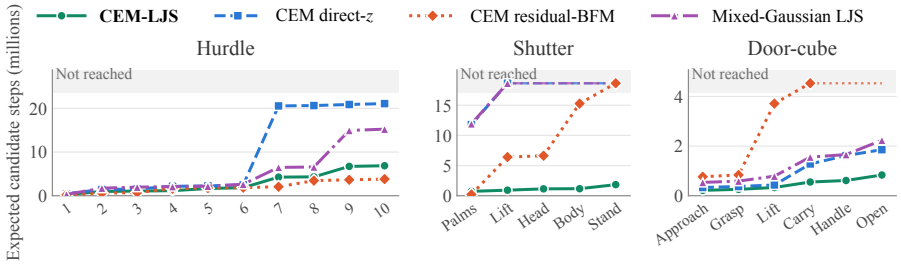


Figure 5: **Search cost to reach each physical effect, ten seeds.** Expected candidate control steps, including unsuccessful attempts (lower is better). Lines entering the "not reached" band mark effects that no run reached.

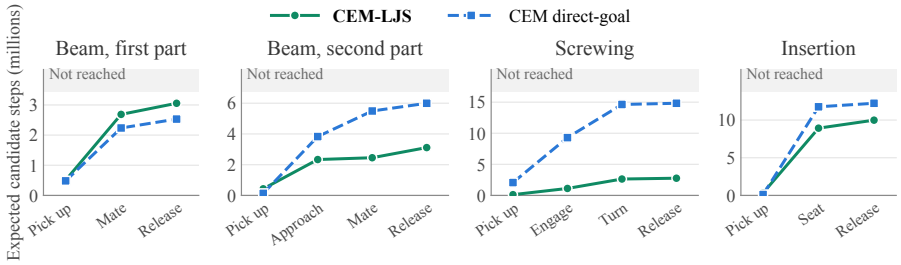


Figure 6: **Search cost to reach each effect with the frozen Play2Perfect policy, ten seeds.** Same measure as Figure 5.

Is the resulting motion physically reasonable? Completing every effect does not tell us whether the motion could be executed on real hardware, since a search can still reach an effect with aggressive

commands (Figure 1). We therefore replay every run with the G1's joint and motor limits and compare CEM-LJS with residual search, which edits the joint targets directly and is the most likely to strain them (Table 3). Comparing the effects that both methods complete, we find that CEM-LJS uses about 40% less mechanical power, has less than half the joint jerk, spends far less time at the motor torque limits, and slips much less under load, and no run of any method exceeds the G1's joint velocity limits. To judge how natural the motion is, we also score it with BFM-Zero's own motion discriminator, which rates CEM-LJS as more human-like even though neither search optimizes for it. This is fairly intuitive, as every LJS proposal is executed through the frozen controller and keeps its coordination. Further, since a task is just a set of physical-effect scores, we can also shape the motion at its roots by adding the discriminator as a cost to the hurdle's scores, which gives a calmer and more human-like crossing with about 30% less mechanical power (Appendix E).

Table 3: **Motion quality of CEM-LJS and CEM residual-BFM** over the 65 effects both complete (medians), with the G1's joint and motor limits. Appendix E gives all 13 measures and the tests.

Measure (per effect)	CEM-LJS	CEM residual-BFM	CEM-LJS better on
Mechanical power (W/kg)	**1.59**	2.61	56/65
Joint jerk, RMS (rad/s^3)	**782**	1660	54/65
Time at the motor torque limit	**0.06**	0.20	43/65
Foot slip under load (m/s)	**0.020**	0.112	55/65
Human-likeness (higher is better)	$-$**0.89**	$-$2.40	57/65

Do the same definitions hold up under changes? We observe that the effect definitions carry over to new starting configurations without any retuning. When the shutter task starts from moved and turned positions, CEM-LJS still gets through from 8 of 10 starts, while no baseline gets through more than twice (Appendix F). For all practical purposes, a single search can settle in a poor local optimum, so we retry a failed effect, from an earlier state if needed, and score every candidate under randomized friction, mass, motor strength, and control delay. CEM-LJS then completes all 30 runs across the three tasks, while the best baseline completes 22 (Table 4 and Appendix F). This largely removes the factor of luck.

Table 4: **Retries with randomized physics, ten seeds per task.** Fully completed tasks and CEM-LJS search time.

Task	CEM-LJS	CEM direct-z	CEM residual-BFM	Mixed-Gaussian LJS	CEM-LJS time
Hurdle (10)	**10/10**	6/10	7/10	2/10	479 s (221–856 s)
Shutter (5)	**10/10**	6/10	0/10	3/10	328 s (137–887 s)
Door-cube (6)	**10/10**	**10/10**	2/10	**10/10**	145 s (128–161 s)
All	**30/30**	22/30	9/30	15/30	

Does the comparison depend on how the effects are written? As a sanity check, we define a task whose definition leaves nothing to tune. The humanoid carries a 3 kg box, held between its palms, along a path of points in 3D space. Each effect only asks the box to reach the next point, and the box must never touch the floor. This minimal definition imposes nothing on the robot or on how it handles the box, and it is the same for every method. With the same retries for every method, CEM-LJS completes the large majority of runs, while no baseline completes more than a fifth of them (Table 5 and Figure 7).

Ablations. To check that the benefit comes from the coupling itself, we vary the optimizer and the proposal space on all three humanoid tasks with ten seeds each. With the same optimizer, proposals pulled back through the Jacobian complete more effects than proposals in the latent space in 11 of the 12 task and optimizer pairs (Appendix D). A random latent subspace of the same dimension as the LJS proposals completes no task. We then ask which part of the coupling matters. Rotating the joint-action axes of CEM-LJS changes little, but searching the Jacobian's own latent directions without the rescaling of LJS completes less than half as many tasks, in line with the analysis in Appendix B. Finally, to make sure that the baselines are not simply mistuned, we also vary their proposal scales on the shutter, and none of them gets through at any scale (Appendix D.1).

Table 5: **Sanity check with a minimal definition**, three paths with five seeds each.

Method	Runs completed	Mean points reached (of 10)
CEM-LJS	**12/15**	**9.1**
CEM direct-z	2/15	2.5
CEM residual-BFM	3/15	4.8
Mixed-Gaussian LJS	1/15	1.5

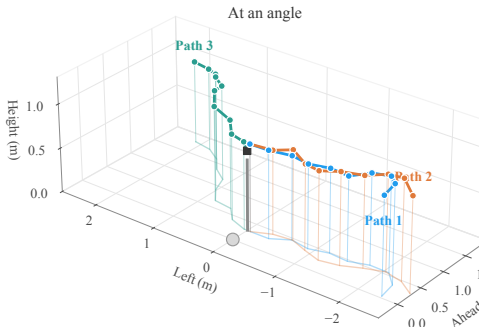


Figure 7: **The three generated paths** for the sanity check.

Table 6: **What makes the coupling work**, ten seeds per task: fully completed tasks and mean accepted effects.

Search	Shutter (5)	Hurdle (10)	Door-cube (6)
CEM-LJS	**7/10**, 3.9	**3/10**, 6.2	**10/10**, 6.0
CEM-LJS, rotated proposal axes	7/10, 3.6	0/10, 5.7	10/10, 6.0
CEM over the Jacobian's latent directions	3/10, 1.6	0/10, 4.1	6/10, 4.8
CEM direct-z	0/10, 0.1	1/10, 4.6	6/10, 4.9
Random 29-D latent subspace	0/10, 0.0	0/10, 0.8	0/10, 1.9

5 Conclusion

We presented Latent Jacobian Steer (LJS), a method for eliciting contact-rich OOD behaviors from frozen latent-conditioned humanoid controllers. LJS proposes physically meaningful changes in joint-action space, pulls them back into the latent space through a damped local inverse of the policy Jacobian, and evaluates the resulting latent commands through simulation. LJS can therefore be interpreted as an interface that makes behavioral foundation models *precise*. Importantly, LJS produces a sequence of latent commands $(z_1, \ldots, z_M)$ rather than an open-loop trajectory of joint actions. While each command is held fixed for a short execution window, the frozen policy continues to receive updated observations and generate new joint targets. These latent sequences can also serve as contact-rich demonstrations for behavioral cloning, reinforcement learning, or policy distillation; initialize subsequent task-specific optimization; and help probe which behaviors are present in a pretrained controller but difficult to access through direct prompting.

The current method is an offline behavior-generation procedure. It requires privileged simulator measurements, saved-state resets, and many parallel candidate rollouts, making it unsuitable for direct real-time deployment. Its success is also limited by the behavioral support of the frozen controller, while the decomposition into physical effects remains human-specified and can admit unintended solutions. A natural next step is to distill successful LJS-generated behaviors into closed-loop policies for efficient execution and eventual sim-to-real evaluation. We hope this simple and elegant method can give rise to new data-generation pipelines as well as learning paradigms that make use of pretrained controllers while not losing the authority of exploring in direct action space.

AI USE STATEMENT

We used generative AI tools to polish manuscript language, assist with literature discovery and summarization, and write and debug research code. AI assistance also supported the translation of author-provided natural-language task specifications into physical-effect measurements, scoring functions, and acceptance conditions, including iterative revisions informed by author feedback and analysis of simulation outputs. The research ideas, problem formulation, and overall research direction originated with the authors.

We reviewed all AI-assisted work, including manuscript text, code, physical-effect definitions, and interpretations of experimental results. The authors retained responsibility for methodological decisions and the claims reported in the paper. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 describe the method, tasks, baselines, and evaluation protocol. Appendix A.2 specifies the CEM updates, Appendix K gives the physical-effect scores and acceptance conditions, and Appendix C lists the seeds, search settings, and the outcome and search time of every evaluation run. All methods use the same finalized task definitions, with the frozen BFM-Zero G1 controller for the humanoid tasks and the frozen Play2Perfect policy for the assembly tasks. The evaluations ran one search at a time on an NVIDIA RTX 4090. Videos of all evaluation runs and the additional results are on the project page, https://latent-jacobian-steer.github.io/, where the code will also be released.

ACKNOWLEDGMENTS

This project was supported by an NUS Seed Grant in Robotics 2025–2029. S K Swaminathan was supported by the International Relations Office of IIT Kharagpur through its semester-away program. We thank Jeric Lew for insightful discussions about the method, and Prof. Aritra Hazra of IIT Kharagpur for supporting and guiding S K Swaminathan throughout the semester-away project, and for his insights.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. {OPAL}: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

Zixuan Chen, Mazeyu Ji, Xuxin Cheng, Xuanbin Peng, Xue Bin Peng, and Xiaolong Wang. Gmt: General motion tracking for humanoid whole-body control. *arXiv:2506.14770*, 2025.

Ziyang Cheng, Tianshu Tang, Jinxin Lan, Xinze Chen, Yuhan Gong, Zhichao Liu, Changzhong Wu, Yahao Mao, Zongyan Deng, Mingxuan Ma, Huasen Xi, Yilong Liu, Yutong Wu, Xiaofeng Wang, Borui Zhang, Bingyao Yu, Yang Wang, Yun Ye, Guan Huang, Xiaojie Jin, Zheng Zhu, and Jiwen Lu. Gigabrain-wbc-0.5: A behavior world model for robust humanoid whole-body tracking with environment interaction, 2026. URL https://arxiv.org/abs/2608.18234.

Alberto Silvio Chiappa, Alessandro Marin Vargas, Ann Zixiang Huang, and Alexander Mathis. Latent exploration for reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Prithwish Dan, Kushal Kedia, Angela Chao, Edward Duan, Maximus Adrian Pace, Wei-Chiu Ma, and Sanjiban Choudhury. X-sim: Cross-embodiment learning via real-to-sim-to-real. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 816–833. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/dan25a.html.

Tony Feng, Trieu H. Trinh, Garrett Bingham, Dawsen Hwang, Yuri Chervonyi, Junehyuk Jung, Joonkyung Lee, Carlo Pagano, Sang hyun Kim, Federico Pasqualotto, Sergei Gukov, Jonathan N.

Lee, Junsu Kim, Kaiying Hou, Golnaz Ghiasi, Yi Tay, YaGuang Li, Chenkai Kuang, Yuan Liu, Hanzhao Lin, Evan Zheran Liu, Nigamaa Nayakanti, Xiaomeng Yang, Heng-Tze Cheng, Demis Hassabis, Koray Kavukcuoglu, Quoc V. Le, and Thang Luong. Towards autonomous mathematics research, 2026. URL https://arxiv.org/abs/2602.10177.

Nikolaus Hansen, Anne Auger, Dimo Brockhoff, Dejan Tusar, and Tea Tušar. COCO: Performance Assessment. ArXiv e-prints, arXiv:1605.03560, May 2016. URL https://inria.hal.science/hal-01315318.

Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobanbabu, Chaoyi Pan, Zeji Yi, Guannan Qu, Kris Kitani, Jessica K. Hodgins, Linxi Fan, Yuke Zhu, Changliu Liu, and Guanya Shi. ASAP: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. In *Robotics: Science and Systems (RSS)*, 2025.

Jiajun Hu, Núria Armengol Urpí, Jin Cheng, and Stelian Coros. Maximum entropy behavior exploration for sim2real zero-shot reinforcement learning. *Reinforcement Learning Journal*, 7, 2026.

Zhi Jing, Siyuan Yang, Jicong Ao, Ting Xiao, Yu-Gang Jiang, and Chenjia Bai. Humanoidgen: Data generation for bimanual dexterous manipulation via LLM reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Mk9ykil8eP.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, 2019. doi: 10.1109/ICRA.2019.8794127.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE J. Robotics Autom.*, 3:43–53, 1987. URL https://api.semanticscholar.org/CorpusID:7687240.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Vince Kurtz. Hydrax: Sampling-based model predictive control on gpu with jax and mujoco mjx, 2024. https://github.com/vincekurtz/hydrax.

Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. Safe mutations for deep and recurrent neural networks through output gradients, 2018. URL https://arxiv.org/abs/1712.06563.

Hongyi Li, Li Peizhuo, Yucheng Tao, Ze Wang, Fangzhou Xu, Jinyi Chen, Yanyan Yuan, Dapeng Jia, Yongbin Jin, Mingfeng Fan, Guillaume Sartoretti, and Hongtao Wang. Pgmt: Perceptive general motion tracking for humanoid robots, 2026a. URL https://arxiv.org/abs/2609.08511.

Jiaman Li, Jiajun Wu, and C Karen Liu. Object motion guided human motion synthesis. *ACM Trans. Graph.*, 42(6), 2023.

Jiaman Li, Alexander Clegg, Roozbeh Mottaghi, Jiajun Wu, Xavier Puig, and Karen Liu. *Controllable Human-Object Interaction Synthesis*, pp. 54–72. 11 2024. ISBN 978-3-031-72939-3. doi: 10.1007/978-3-031-72940-9_4.

Yitang Li, Zhengyi Luo, Tonghe Zhang, Cunxi Dai, Anssi Kanervisto, Andrea Tirinzoni, Haoyang Weng, Kris Kitani, Mateusz Guzek, Ahmed Touati, Alessandro Lazaric, Matteo Pirotta, and Guanya Shi. BFM-zero: A promptable behavioral foundation model for humanoid control using unsupervised reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=jkhl2oI0g5.

Qiayuan Liao, Takara E. Truong, Xiaoyu Huang, Yuman Gao, Guy Tevet, Koushil Sreenath, and C. Karen Liu. Beyondmimic: From motion tracking to versatile humanoid control via guided diffusion. *Science Robotics*, 11(117):eadx8924, 2026. doi: 10.1126/scirobotics.adx8924. URL https://www.science.org/doi/abs/10.1126/scirobotics.adx8924.

Junjie Lu, Xinyao Qin, Yuhua Jiang, Kaixin Wang, Chuheng Zhang, Bin Liang, Jun Yang, Min Xu, and Li Zhao. Unisteer: Unified noise steering for efficient human-guided vla adaptation, 2026. URL https://arxiv.org/abs/2605.10821.

Tyler Ga Wei Lum, Kushal Kedia, Karen Liu, and Jeannette Bohg. Play2perfect: What matters in dexterous play pretraining for precise assembly? In *Conference on Robot Learning (CoRL)*, 2026.

Zhengyi Luo, Ye Yuan, Tingwu Wang, Chenran Li, Fernando Castañeda, Sirui Chen, Zi-Ang Cao, Jiefeng Li, David Minor, Qingwei Ben, Jinhyung Park, David Sami, Zi Wang, Xingye Da, Runyu Ding, Cyrus Hogg, Lina Song, Edy Lim, Eugene Jeong, Tairan He, Haoru Xue, Wenli Xiao, Simon Yuen, Jan Kautz, Yan Chang, Umar Iqbal, Linxi "Jim" Fan, and Yuke Zhu. Sonic: Supersizing motion tracking for natural humanoid whole-body control. *Science Robotics*, 11(117): eaed4592, 2026. doi: 10.1126/scirobotics.aed4592. URL https://www.science.org/doi/abs/10.1126/scirobotics.aed4592.

Mayank Mittal, Pascal Roth, James Tigue, Antoine Richard, Octi Zhang, Peter Du, Antonio Serrano-Muñoz, Xinjie Yao, René Zurbrügg, Nikita Rudin, et al. Isaac lab: A GPU-accelerated simulation framework for multi-modal robot learning. *arXiv preprint arXiv:2511.04831*, 2025. URL https://arxiv.org/abs/2511.04831.

Tongzhou Mu, Minghua Liu, and Hao Su. Drs: Learning reusable dense rewards for multi-stage tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6CZ50WgfCG.

Michael Murray, Daphne Chen, Simran Bagaria, Dean Fortier, Tess Hellebrekers, Galen Mullins, Harshavardhan Gajarla, Oier Mees, Maya Cakmak, and Andrey Kolobov. FlowDAgger: Human-in-the-loop adaptation of generative robot policies in latent space. In *Proceedings of the 10th Conference on Robot Learning*, 2026.

Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. In *Forty-first International Conference on Machine Learning*, 2024a. URL https://openreview.net/forum?id=LhNsSaAKub.

Seohong Park, Oleh Rybkin, and Sergey Levine. METRA: Scalable unsupervised RL with metric-aware abstraction. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=c5pwL0Soay.

Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4):1–17, July 2022. ISSN 1557-7368. doi: 10.1145/3528223.3530110. URL http://dx.doi.org/10.1145/3528223.3530110.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020.

Cristina Pinneri, Shambhuraj Sawant, Sebastian Blaes, Jan Achterhold, Joerg Stueckler, Michal Rolinek, and Georg Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning 2020*, 2020. URL https://corlconf.github.io/paper_217.

Michael Posa, Cecilia Cantu, and Russ Tedrake. A Direct Method for Trajectory Optimization of Rigid Bodies Through Contact. *International Journal of Robotics Research (IJRR)*, 33(1):69–81, January 2014. doi: 10.1177/0278364913506757. URL http://ijr.sagepub.com/content/33/1/69.short.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth exploration for robotic reinforcement learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann (eds.), *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pp.

1634–1644. PMLR, 08–11 Nov 2022. URL https://proceedings.mlr.press/v164/raffin22a.html.

Nemanja Rakicevic, Antoine Cully, and Petar Kormushev. Policy manifold search: exploring the manifold hypothesis for diversity-based neuroevolution. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, pp. 901–909. ACM, 2021. doi: 10.1145/3449639.3459320. URL http://dx.doi.org/10.1145/3449639.3459320.

Nadun Ranawaka, Josiah Wong, Wei-Lin Pai, Wei-Teng Chu, Tianyuan Dai, Masoud Moghani, Hang Yin, Yunfan Jiang, Wesley Durbano, Brandon Huynh, Yu Fang, Danfei Xu, Ruohan Zhang, Li Fei-Fei, Linxi Fan, Bowen Wen, Ajay Mandlekar, and Yuke Zhu. Simfoundry: Modular and automated scene generation for policy learning and evaluation, 2026. URL https://arxiv.org/abs/2606.28276.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999. doi: 10.1023/A:1010091220143.

Pierre Schumacher, Daniel F.B. Haeufle, Dieter Büchler, Syn Schmitt, and Georg Martius. Dep-rl: Embodied exploration for reinforcement learning in overactuated and musculoskeletal systems. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, May 2023. URL https://openreview.net/forum?id=C-xa_D3oTj6.

Carmelo Sferrazza, Dun-Ming Huang, Xingyu Lin, Youngwoon Lee, and Pieter Abbeel. Humanoidbench: Simulated humanoid benchmark for whole-body locomotion and manipulation. 07 2024. doi: 10.15607/RSS.2024.XX.061.

Lucy Xiaoyang Shi, Joseph J Lim, and Youngwoon Lee. Skill-based model-based reinforcement learning. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 2262–2272. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/shi23a.html.

Andy Tang, William Chen, Andrew Wagenmaker, Chelsea Finn, and Sergey Levine. Improving robotic generalist policies via flow reversal steering. In *3rd Workshop on Semantic Reasoning and Goal Understanding in Robotics (RSS 2026)*, 2026. URL https://openreview.net/forum?id=lZLHs6P90d.

Ilyass Taouil, Michal Ciebelski, Shafeef Omar, Haizhou Zhao, Angela Dai, Aaron M. Johnson, and Majid Khadiv. MotionDisco: Motion discovery for extreme humanoid loco-manipulation, 2026. URL https://arxiv.org/abs/2606.06139.

Andrea Tirinzoni, Ahmed Touati, Jesse Farebrother, Mateusz Guzek, Anssi Kanervisto, Yingchen Xu, Alessandro Lazaric, and Matteo Pirotta. Zero-shot whole-body humanoid control via behavioral foundation models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9sOR0nYLtz.

Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=2a96Bf7Qdrg.

Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *International Joint Conference on Artificial Intelligence*, 2015. URL https://api.semanticscholar.org/CorpusID:11309330.

Charles W. Wampler. Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16(1):93–101, 1986. doi: 10.1109/TSMC.1986.289285.

Max Wilcoxson, Qiyang Li, Kevin Frans, and Sergey Levine. Leveraging skills from unlabeled prior data for efficient online exploration. In *International Conference on Machine Learning (ICML)*, 2025.

Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control: From theory to parallel computation. *Journal of Guidance, Control, and Dynamics*, 40:344–357, 01 2017. doi: 10.2514/1.G001921.

Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4974–4981, 2025. doi: 10.1109/ICRA55743.2025.11127320.

Ruiqi Yu, Yiwen Wang, Yuan Hao, Jun Wu, and Qiuguo Zhu. SSR: Scaling surefooted and symmetric humanoid traversal toward the open world. In *10th Conference on Robot Learning (CoRL)*, 2026a. URL https://arxiv.org/abs/2605.30770.

Runyi Yu, Xiaoyi Lin, Ji Ma, Yinhuai Wang, Koukou Luo, Jiahao Ji, Huayi Wang, Wenjia Wang, Runhan Zhang, Ping Tan, Ting Wu, Ruoli Dai, Qifeng Chen, and Lei Han. Omnicontact: Chaining meta-skills via contact flow for generalizable humanoid loco-manipulation, 2026b. URL https://arxiv.org/abs/2606.26201.

Yanjie Ze, Zixuan Chen, Joao Pedro Araujo, Zi-ang Cao, Xue Bin Peng, Jiajun Wu, and Karen Liu. Twist: Teleoperated whole-body imitation system. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2143–2154. PMLR, 27–30 Sep 2025. URL https://proceedings.mlr.press/v305/ze25a.html.

A Theory

A.1 Forward-backward representations and BFM-Zero

We briefly describe the forward-backward (FB) representation underlying BFM-Zero (Li et al., 2026b). For simplicity, we write the expressions in terms of the state s, suppressing the distinction between privileged state and observation history.

The FB representation learns a forward mapping F and a backward mapping B that together approximate the discounted future-state occupancy under the latent-conditioned policy π_z:

$$M^{\pi_z}(s, u, \mathrm{d}s') \approx F(s, u, z)^\top B(s')\, \rho(\mathrm{d}s'),$$

where ρ is the reference state distribution and

$$M^{\pi_z}(s, u, X) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbf{1}\{s_{t+1} \in X\} \,\middle|\, s_0 = s,\ u_0 = u,\ \pi_z\right].$$

Thus, F and B describe which future states a latent-conditioned policy is expected to visit.

For a state-based reward r, using the convention that rewards are evaluated at the next state, this factorization gives

$$Q_r^{\pi_z}(s, u) \approx F(s, u, z)^\top \mathbb{E}_{s' \sim \rho}[B(s') r(s')]\,.$$

The policy is trained to favor actions with large $F(s, u, z)^\top z$. Consequently, the reward-weighted backward features provide a latent command for a downstream reward:

$$z_r = \mathbb{E}_{s \sim \rho}[B(s) r(s)]\,.$$

This provides a way to prompt the pretrained policy without task-specific retraining. Similarly, a desired state can be embedded through B, while a reference motion can be represented by a sequence of state embeddings. The resulting commands are normalized to the controller's latent domain before execution.

These prompting mechanisms explain the behavioral structure of the pretrained controller. LJS instead searches latent commands through the frozen actor's Jacobian and simulator-evaluated physical effects; it does not use F or B during search. Reward prompting also cannot express our tasks. The backward map B of the G1 controller takes only the robot's own state, namely its joint positions and velocities and the positions, orientations, and velocities of its bodies, so a reward on the shutter, the cube, the door, or contact forces is not a function of its input. A prompted latent is also a single command for the whole task, while our tasks are ordered sequences of effects.

A.2 CEM UPDATES FOR JOINT-ACTION SEARCH

We adapt the CEM implementation in Hydrax (Kurtz, 2024) to search over joint-action perturbations. At each planning step, we initialize

$$\mu_0 = 0, \qquad s_0 = \sigma \mathbf{1}_{29},$$

where $\sigma > 0$ is the initial exploration scale. We divide the candidate budget across R rounds. In round q, we draw N_q proposals and map them into latent commands:

$$\Delta u_i^{(q)} \sim \mathcal{N}\big(\mu_q, \mathrm{diag}(s_q^2)\big), \qquad z_i^{(q)} = \Pi\Big(z + A\Delta u_i^{(q)}\Big).$$

One proposal is set to the current mean, $\Delta u_1^{(q)} = \mu_q$, so the first round includes the unchanged command. The latent anchor z and pull-back map A remain fixed throughout these rounds.

We simulate each candidate and rank admissible proposals by their physical-effect scores. Let $\mathcal{V}_q$ denote the admissible candidates. The elite set $\mathcal{E}_q$ contains the lowest-scoring

$$n_q^{\mathrm{elite}} = \min(|\mathcal{V}_q|, \max(2, \lceil \kappa N_q \rceil))$$

proposals, where $\kappa \in (0, 1]$ is the elite fraction. When the elite set is nonempty, we update

$$\mu_{q+1} = \frac{1}{|\mathcal{E}_q|} \sum_{i \in \mathcal{E}_q} \Delta u_i^{(q)},$$

$$s_{q+1,j} = \max\left(s_{\min}, \sqrt{\frac{1}{|\mathcal{E}_q|} \sum_{i \in \mathcal{E}_q} \left(\Delta u_{i,j}^{(q)} - \mu_{q+1,j}\right)^2}\right), \qquad j = 1, \ldots, 29.$$

The floor $s_{\min} > 0$ prevents the proposal distribution from collapsing completely. If no candidate is admissible, we retain the previous distribution:

$$\mu_{q+1} = \mu_q, \qquad s_{q+1} = s_q.$$

After all rounds, we select the lowest-scoring admissible candidate evaluated across the entire planning step. Its latent command is executed for a short window before recomputing the Jacobian and restarting CEM from the resulting state. If no admissible candidate was found, the search terminates without declaring task completion.

B WHY DOES LJS WORK, MATHEMATICALLY?

A small change in the latent command can move some joints strongly and others barely at all. LJS accounts for this unequal sensitivity when constructing its proposals. Intuitively, if we choose some random (including 29-D) space over the action space itself, some degrees of freedom might be unfairly disadvantaged. This is why we argue that LJS gives exploration enough authority to discover precise changes.

At the current observation s and latent command z, let

$$J = \frac{\partial \pi(s, z)}{\partial z}.$$

Locally, a latent change Δz produces the action change $\Delta u \approx J\Delta z$. LJS samples a desired action change δu and solves

$$\min_{\Delta z} \|J\Delta z - \delta u\|^2 + \lambda^2 \|\Delta z\|^2.$$

Its solution is

$$\Delta z = J^\top (JJ^\top + \lambda^2 I)^{-1} \delta u.$$

Thus, LJS tries to realize the proposed action change without requiring an excessively large latent change.

Why is this different from searching in another 29-dimensional space? Consider even the latent directions selected by the Jacobian itself. Write $J = U\Sigma V^\top$, with nonzero singular values s_i. Sampling $\Delta z = V\epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$, produces

$$J\Delta z = U\Sigma\epsilon.$$

The action variance along direction i is $\sigma^2 s_i^2$. Consequently, some action directions receive much less exploration than others, despite equal sampling variance in the search space.

With LJS and $\delta u \sim \mathcal{N}(0, \sigma^2 I)$, that variance is instead

$$\sigma^2 \left(\frac{s_i^2}{s_i^2 + \lambda^2} \right)^2.$$

For $s_i \gg \lambda$, it is approximately σ^2: LJS compensates for the controller's unequal sensitivities. For $s_i \ll \lambda$, damping suppresses the correction rather than demanding a large latent displacement. These action directions can be combinations of joints, not necessarily individual joints.

The benefit is therefore more than reducing the search dimension. LJS makes comparable proposals produce approximately comparable immediate action changes in directions the controller can realize. An arbitrary low-dimensional latent space need not do this. Appendix B.1 tests this experimentally.

B.1 What makes the coupling work

Appendix B explains why LJS works. The policy responds very strongly to some latent directions and barely to others, so equally sized random changes in the latent command mostly move the joints along a few directions. The damped inverse of LJS corrects for this, so that equally sized joint-action proposals produce comparably sized changes in the joints. Two simpler explanations could also account for the results of Table 10, and we test both on all three tasks with the protocol of Appendix C. First, CEM adapts its search width separately along each axis it samples in, so LJS might help simply because the joint axes happen to be convenient axes for CEM. To test this, we rotate the joint-action proposals of CEM-LJS by one fixed random rotation. This keeps the initial proposals the same and only changes the axes along which CEM adapts. Second, LJS might help only because it searches the 29 latent directions that the policy actually responds to. To test this, we let CEM search along exactly these directions, the right-singular vectors of the policy Jacobian ($\Delta z = Vc$), but without the rescaling of LJS.

With rotated axes, CEM-LJS completes as many shutter and door-cube tasks as before and a similar number of hurdle effects, although none of its hurdle runs reaches the final effects (Table 6). Over the three tasks, it completes 17 of 30 runs against 20 for CEM-LJS (Fisher's exact test, $p = 0.60$). Searching along the Jacobian's latent directions without the rescaling behaves very differently. It completes only 9 of 30 runs ($p = 0.009$) and fewer effects on every task, though still far more than a random 29-dimensional latent subspace, which completes none. A natural suspicion is that its proposals are simply too small. Scaling them up until they move the joints as much as CEM-LJS proposals, however, still leaves it at 3 of 10 shutters. Together, these tests point to the two parts of LJS described above: the Jacobian selects the latent directions the policy responds to, and the damped inverse balances how strongly each of them moves the joints.

C Evaluation with ten seeds

Seeds and settings. The humanoid results use the seeds 12, 10, 19, 2027, 2004, 1264, 2681, 752, 2360, and 2981. None of them was used while defining the physical effects. All four methods use the same effects and the same search budget on every task: candidates are simulated for 40 control steps, the selected command is executed for 20 steps, each decision evaluates 1,024 candidates (1,023 sampled and the unchanged command) over four CEM rounds, and each effect allows at

most 30 decisions. Because the selected candidate has already been simulated, we continue from its simulated state at the end of the execution window instead of simulating it again. When a CEM round contains no candidate that satisfies the hard constraints of the current effect, the losses of those constraints guide the sampling of the next round, and a candidate is only executed if it satisfies every hard constraint. Each method uses the same proposal scales on every seed. Every search runs alone on an NVIDIA RTX 4090 in a fresh process.

Search time. Table 7 gives the wall-clock search time of the successful runs. The search time and outcome of every run are listed in Table 9, and the videos of all evaluation runs are available on the project page.

Table 7: **Wall-clock search time of successful runs, ten seeds.** Mean (fastest), on an NVIDIA RTX 4090 with one search at a time. A dash marks a method that never completes the task.

Method	Hurdle	Shutter	Door-cube
CEM-LJS	137 s (104 s)	77 s (54 s)	61 s (55 s)
CEM direct-z	152 s (152 s)	–	72 s (58 s)
CEM residual-BFM	89 s (80 s)	–	–
Mixed-Gaussian LJS	189 s (189 s)	–	82 s (62 s)

Statistical comparison. To check that the differences in Table 1 are not seed-to-seed noise, we compare the accepted effects of CEM-LJS with each baseline over the ten seeds with a two-sided Mann–Whitney test (Table 8). On the shutter and door-cube, CEM-LJS differs from every baseline. On the hurdle, the methods overlap more, and CEM-LJS has the highest mean.

Table 8: **CEM-LJS against each baseline, ten seeds.** Two-sided Mann–Whitney p-values on the accepted effects per seed.

CEM-LJS vs.	Hurdle	Shutter	Door-cube
CEM direct-z	0.29	**0.001**	**0.034**
CEM residual-BFM	0.70	**0.020**	**<0.001**
Mixed-Gaussian LJS	0.12	**0.001**	**0.015**

Table 9: **Wall-clock search time and outcome of every BFM-Zero evaluation run, ten seeds.** Each entry gives the accepted effects / task total and, in parentheses, the search time in seconds on an NVIDIA RTX 4090. Fully completed tasks are in bold.

Task	Method	12	10	19	2027	2004	1264	2681	752	2360	2981
Hurdle	CEM-LJS	1/10 (88)	8/10 (162)	6/10 (157)	**10/10 (104)**	4/10 (104)	**10/10 (110)**	1/10 (88)	**10/10 (198)**	6/10 (78)	6/10 (119)
	CEM direct-z	6/10 (175)	1/10 (45)	**10/10 (152)**	6/10 (98)	6/10 (136)	3/10 (113)	6/10 (142)	1/10 (108)	6/10 (175)	1/10 (35)
	CEM residual-BFM	**10/10 (93)**	1/10 (34)	**10/10 (94)**	7/10 (88)	3/10 (115)	3/10 (124)	1/10 (31)	7/10 (80)	**10/10 (80)**	1/10 (20)
	Mixed-Gaussian LJS	1/10 (43)	6/10 (130)	1/10 (51)	1/10 (46)	1/10 (49)	6/10 (152)	1/10 (48)	8/10 (244)	**10/10 (189)**	1/10 (58)
Shutter	CEM-LJS	4/5 (103)	**5/5 (86)**	**5/5 (105)**	**5/5 (61)**	0/5 (86)	**5/5 (75)**	**5/5 (75)**	0/5 (84)	**5/5 (54)**	**5/5 (83)**
	CEM direct-z	1/5 (122)	0/5 (79)	0/5 (80)	0/5 (83)	0/5 (80)	0/5 (82)	0/5 (83)	0/5 (80)	0/5 (78)	0/5 (80)
	CEM residual-BFM	1/5 (91)	1/5 (103)	1/5 (97)	1/5 (98)	1/5 (97)	4/5 (198)	1/5 (96)	1/5 (91)	3/5 (156)	1/5 (94)
	Mixed-Gaussian LJS	0/5 (77)	0/5 (76)	0/5 (75)	0/5 (77)	0/5 (79)	0/5 (73)	0/5 (80)	0/5 (76)	1/5 (106)	0/5 (72)
Door-cube	CEM-LJS	**6/6 (60)**	**6/6 (55)**	**6/6 (57)**	**6/6 (58)**	**6/6 (55)**	**6/6 (56)**	**6/6 (71)**	**6/6 (55)**	**6/6 (70)**	**6/6 (69)**
	CEM direct-z	3/6 (107)	**6/6 (69)**	3/6 (49)	**6/6 (78)**	4/6 (81)	3/6 (112)	**6/6 (66)**	**6/6 (79)**	**6/6 (58)**	**6/6 (82)**
	CEM residual-BFM	2/6 (109)	3/6 (33)	2/6 (92)	0/6 (87)	3/6 (35)	0/6 (85)	2/6 (96)	3/6 (34)	2/6 (104)	0/6 (80)
	Mixed-Gaussian LJS	3/6 (39)	**6/6 (67)**	0/6 (75)	5/6 (84)	3/6 (107)	**6/6 (82)**	**6/6 (62)**	**6/6 (86)**	**6/6 (114)**	0/6 (81)

D SEARCH ABLATIONS

Ten seeds on all three tasks. We run every search variant under the protocol of Appendix C, with the same seeds, budgets, and effect definitions. Only the proposal coordinates, the population update, or the mix of proposals change between variants (Table 10).

The coupling helps across optimizers. CEM-LJS completes each task most often. With the same optimizer (CEM, MPPI, Gaussian, or targeted refinement) on the same task, proposals pulled back through the Jacobian complete more effects than proposals in the latent space in 11 of the 12 pairs.

Table 10: **Search ablations, ten seeds per task.** Fully completed tasks and mean accepted effects. The first block searches joint-action proposals pulled back through the Jacobian, and the second searches the latent command directly.

Search	Shutter (5)	Hurdle (10)	Door-cube (6)
CEM-LJS	**7/10**, 3.9	**3/10**, 6.2	**10/10**, 6.0
MPPI LJS	4/10, 2.0	2/10, 6.3	9/10, 5.4
LATTICE-style LJS	3/10, 1.5	1/10, 3.6	7/10, 4.5
Zero-mean refinement LJS	2/10, 1.0	1/10, 4.4	2/10, 2.1
Targeted-mean refinement LJS	1/10, 0.5	0/10, 3.6	4/10, 4.2
Gaussian-only LJS	0/10, 0.1	1/10, 5.5	2/10, 3.0
CEM direct-z	0/10, 0.1	1/10, 4.6	6/10, 4.9
MPPI direct-z	3/10, 1.5	0/10, 2.3	6/10, 4.8
Gaussian direct-z	0/10, 0.0	0/10, 2.7	4/10, 4.4
Targeted refinement direct-z	0/10, 0.0	0/10, 1.6	3/10, 3.6
Random 29-D latent subspace	0/10, 0.0	0/10, 0.8	0/10, 1.9

The one exception is the Gaussian search on door-cube. The difference is largest with CEM and smaller with MPPI. A random 29-dimensional latent subspace, which matches the dimension of the LJS proposals, completes no task.

The optimizer still matters within LJS. Among the LJS variants, CEM completes the most tasks, and MPPI comes close on hurdle and door-cube. Variants that rely on Gaussian proposals or on targeted refinement complete far fewer tasks, especially on the shutter, where the hands have to be placed precisely beneath the panel.

D.1 Sensitivity to the proposal settings

To check that the comparison does not hinge on the proposal settings of each method, we rerun the main methods on the shutter, where they differ most, with their proposal scale lowered and raised, and CEM-LJS also with its damping λ lowered and raised. Everything else follows Appendix C (Table 11). Neither baseline gets through the shutter at any of its settings, while CEM-LJS completes it at every setting, most often at its default setting and with the damping quartered. Almost all of its failures at the other settings stop at the first effect, where both palms must be placed beneath the panel. The damping explains much of this. Along the CEM-LJS shutter runs, the policy Jacobian has a median singular value of 0.048, and the damped inverse realizes at least half of a proposed joint change along a median of 24 of the 29 directions with the default $\lambda = 0.02$ and 26 with $\lambda \times 0.25$, but only 9 with $\lambda \times 4$. With four times the damping, most of a proposed joint change no longer reaches the command, and the search completes 2 of 10 shutters, the same as with half the proposal scale.

Table 11: **Shutter with changed proposal settings**, ten seeds per row. Fully completed tasks and mean accepted effects. The first row of each block is the default setting.

Search	Setting	Shutter (5)
CEM-LJS	default	**7/10**, 3.9
	scale $\times 0.5$	2/10, 1.2
	scale $\times 2$	5/10, 2.9
	damping $\lambda \times 0.25$	7/10, 3.5
	damping $\lambda \times 4$	2/10, 1.0
CEM direct-z	default	0/10, 0.1
	scale $\times 0.3$	0/10, 0.0
	scale $\times 3$	0/10, 0.1
CEM residual-BFM	default	0/10, 1.5
	scale $\times 0.5$	0/10, 1.2
	scale $\times 2$	0/10, 1.0

D.2 Latent proposal scale, PPO, and SONIC on a single shutter seed

On a single shutter seed, with the first four shutter effects and a total budget of 1,310,720 candidate control steps, we also test the scale of direct latent proposals, policy learning with PPO, and SONIC as the pretrained controller (Table 12).

Table 12: **PPO and SONIC on a single shutter seed, first four effects.** CEM-LJS with BFM-Zero on the same seed is given for reference. Search cost counts simulated candidate control steps, and successful searches stop early. PPO counts training control steps and is not directly comparable. The SONIC entry excludes later diagnostic searches.

Variant	Effects	Control steps
CEM-LJS (BFM-Zero)	**4/4**	**696,320**
PPO direct-z	0/4	1,310,720 training
PPO through LJS	0/4	1,310,720 training
Mixed-Gaussian LJS with SONIC	1/4	696,320

Latent proposal scale. Direct latent Gaussian scales of 0.15, 0.5, and 1.5 all fail at hand placement under matched budgets, indicating that perturbation magnitude alone does not resolve the precision problem.

Using the score as an RL reward. PPO trained with the negative physical-effect score completes no effect within this budget, either through direct latent commands or through LJS-coupled proposals. The Jacobian mapping alone therefore does not make this objective easy to optimize through policy learning. These training interactions differ from candidate-rollout evaluations, so the two budgets are not directly comparable.

SONIC as the pretrained controller. Replacing BFM-Zero with SONIC (Luo et al., 2026) does not get beyond the first effect. We tried quantized FSQ tokens and continuous decoder inputs, mixed-Gaussian and CEM search, several noise scales and damping values, and longer horizons. The best run raises the shutter to $0.415\,\mathrm{m}$ of the required $0.75\,\mathrm{m}$, collapses into a low, unstable posture, and never crosses. SONIC tracks supplied motions very well, but in our experiments its inputs could not be steered toward this interaction from the physical-effect scores.

E Motion quality

Measurements. We replay every recorded control step of the ten-seed runs in MuJoCo with BFM-Zero's PD law, the joint position and velocity limits of the Unitree G1, and the motor torque limits of the scene. Human-likeness is the logit of BFM-Zero's own motion discriminator, with the latent inferred from each run's states. Neither method optimizes it. We compare CEM-LJS with CEM residual-BFM over the 65 effects that both complete on the same task and seed, using medians and a two-sided Wilcoxon signed-rank test (Table 13). CEM-LJS is better on 12 of the 13 measures. No run of any method exceeds a G1 joint velocity limit. The commanded torque does peak above the motor limit for every method (median peak of 1.3 to 2.1 times the limit per task). Because this also happens for CEM direct-z, which only changes the latent command of BFM-Zero, the peaks come from BFM-Zero's own PD tracking during contact rather than from LJS. Whenever the commanded torque exceeds a motor's limit, the motor simply applies its maximum torque instead, as a real motor would. Over the 75 effects that both complete in the runs with retries (Appendix F), CEM-LJS is better on all 13 measures and uses about half the mechanical power.

Shaping the motion within the search. Since a task is just a set of physical-effect scores, we can add a cost for motion quality to the scores without changing anything else. We add the excess adversarial cost of BFM-Zero's motion discriminator as a cost-only term to every hurdle effect. It never enters the acceptance conditions. We run the search with retries (Appendix F) on ten seeds, with and without this term, and compare the 79 effects completed in both (Table 14). The crossing becomes calmer and more human-like on most effects. The search becomes longer, and two of the ten seeds reach the time limit.

Table 13: **Motion quality of CEM-LJS and CEM residual-BFM** over the 65 effects that both complete on the same task and seed (medians). The last two columns give the number of effects on which CEM-LJS is better and the Wilcoxon signed-rank p-value.

Measure (per effect)	CEM-LJS	CEM residual-BFM	CEM-LJS better	p
Mechanical power (W/kg)	**1.59**	2.61	56/65	6×10^{-7}
Joint jerk, RMS (rad/s^3)	**782**	1660	54/65	1×10^{-7}
Peak commanded torque / motor limit	**1.28**	2.53	54/65	4×10^{-8}
Time at the motor torque limit (fraction)	**0.06**	0.20	43/65	9×10^{-3}
Joint-target rate, RMS (rad/s)	**2.23**	4.03	55/65	1×10^{-7}
Peak base angular speed (rad/s)	**2.93**	4.05	58/65	3×10^{-10}
Peak joint speed / G1 limit	**0.33**	0.50	54/65	5×10^{-8}
Excursion beyond joint range (rad)	**0.005**	0.025	55/65	1×10^{-9}
Foot slip under load (m/s)	**0.020**	0.112	55/65	2×10^{-8}
Fraction of time with self-contact	**0.24**	0.42	44/65	2×10^{-4}
Deepest interpenetration (m)	**0.008**	0.011	41/65	2×10^{-2}
Peak contact force (body weights)	3.03	3.47	35/65	0.3
Human-likeness (higher is better)	-0.89	-2.40	57/65	5×10^{-10}

Table 14: **Hurdle with BFM-Zero's discriminator added to the effect scores**, over the 79 effects completed with and without it on the same seed. Ratios are medians of with / without.

Measure	With / without	Lower with discriminator	p
Mechanical power	**0.71**	54/79	1×10^{-4}
Joint jerk, RMS	**0.80**	49/79	7×10^{-4}
Joint-target rate, RMS	**0.84**	55/79	3×10^{-4}
Peak base angular speed	**0.79**	54/79	4×10^{-4}
Peak joint speed / G1 limit	**0.89**	49/79	1×10^{-2}
RMS torque / motor limit	**0.92**	51/79	3×10^{-2}
Deepest interpenetration	**0.80**	48/79	2×10^{-2}
Human-likeness higher with discriminator		67/79 effects ($p = 2 \times 10^{-11}$)	
Full hurdle completed		8/10 with, 10/10 without	
Median search time		688 s with, 461 s without	

F Robustness to new starts and varied physics

Moved and turned starts. We start the shutter task from ten configurations in which the robot is both moved (0.05–0.19 m, drawn uniformly within 0.20 m) and turned ($-11.2°$ to $+11.8°$, drawn uniformly within $\pm 15°$). The joints and the shutter are unchanged, and so are the effect definitions. The final standing effect asks the robot to face through the doorway, which for the default start is the same as its starting heading. Each start uses one new seed and the settings of Appendix C (Table 15). CEM-LJS gets through from 8 of the 10 starts, more than any baseline (Fisher's exact test, $p \leq 0.023$ against each baseline).

Table 15: **Shutter from ten moved and turned starts.** Effects completed from each start, the number of fully completed tasks, and the wall-clock search time of successful runs, mean (fastest).

Method	Effects per start	Full shutter	Search time
CEM-LJS	0, 5, 5, 5, 5, 1, 5, 5, 5, 5	**8/10**	84 s (57 s)
CEM direct-z	0, 0, 5, 0, 0, 0, 0, 0, 2, 0	1/10	158 s (158 s)
CEM residual-BFM	1, 1, 1, 1, 4, 1, 1, 1, 1, 1	0/10	–
Mixed-Gaussian LJS	0, 0, 0, 0, 0, 5, 5, 0, 0, 0	2/10	95 s (93 s)

Retries and randomized physics. A CEM search can settle in a poor local optimum, for example when a hurdle effect is accepted in a pose that makes the next effect hard to reach. For practical

use, we add two changes that apply to every method. First, a failed effect is searched again from the same state with a new seed, and if it fails again, the previous effect is searched again so that the failing effect starts from a different state, up to six times per run. Second, every candidate is scored under four physics settings drawn from ranges used in humanoid sim-to-real work (He et al., 2025; Yu et al., 2026a): friction between 0.2 and 1.1, link masses within $\pm 10\%$, PD gains between -7.5% and $+5\%$ (P) and within $\pm 10\%$ (D), and a control delay of 20 to 40 ms. Each run is limited to 15 minutes. We use ten new seeds, 20 to 29, and each method keeps its own proposal settings and the budgets of Appendix C (Table 4). CEM-LJS completes all 30 runs, while every baseline fails on some of them (Fisher's exact test, $p \leq 0.005$ against each baseline). The retries largely remove the factor of luck from a single search, and the remaining trade-off is between robustness and wall-clock time.

G Sanity check with minimal definitions

Task. The humanoid stands with a 0.20 m, 3 kg box resting on a thin post between its palms. The controller does not squeeze an object on its own, and without the post the box falls within about 0.1 s, so the search has to make the robot grip the box, lift it off the post, and carry it while walking. The task has ten effects, one for each point of a path. Effect k has a single requirement, $\|p_{\text{box}} - q_k\|_2 \leq 0.05\,\text{m}$, scored by the squared distance between the box center and the point q_k (Appendix K.2.4). One constraint holds throughout every effect: the box never touches the floor. Nothing else is measured, so there are no contact, force, or posture terms that could be tuned toward a method.

Paths. We generate three paths with a fixed rule. Each has ten points, 0.30 m apart horizontally, at box-center heights between 0.94 and 1.18 m. The first point lifts the box off the post, after which the path heads to the left or right and turns by up to $45°$ at each point. Paths that come back near the post, cross themselves, turn less than $90°$ in total, or vary less than 0.15 m in height are drawn again. None of these paths was used while writing the task (Figure 8).

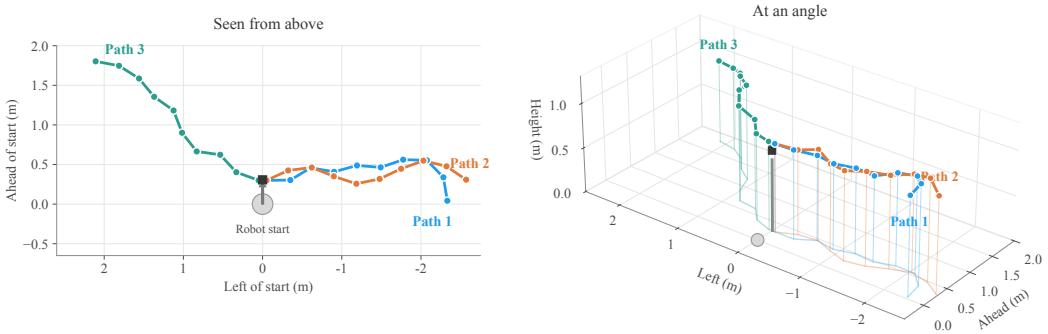


Figure 8: **The three generated paths** for the sanity check, seen from above and at an angle.

Results. Every method runs five seeds on each path with retries and without randomized physics, a 15-minute limit per run, and the budgets of Appendix C. A run counts as complete when the box reaches all ten points in order without touching the floor, which we check by replaying the recorded states (Table 16 and Figure 9). CEM-LJS completes 12 of the 15 runs, against at most 3 for any baseline (Fisher's exact test, $p \leq 0.003$ against each baseline).

H Assembly with the frozen Play2Perfect policy

Controller and tasks. Play2Perfect (Lum et al., 2026) pretrains a goal-conditioned policy for a Sharpa hand on a Franka arm through play, and then fine-tunes it separately for each assembly task.

Table 16: **Sanity check with minimal definitions.** Completed runs per path (five seeds each) and the mean number of points reached.

Method	Path 1	Path 2	Path 3	All	Points reached (of 10)
CEM-LJS	**4/5**	**4/5**	**4/5**	**12/15**	**9.1**
CEM direct-z	1/5	1/5	0/5	2/15	2.5
CEM residual-BFM	2/5	0/5	1/5	3/15	4.8
Mixed-Gaussian LJS	1/5	0/5	0/5	1/15	1.5

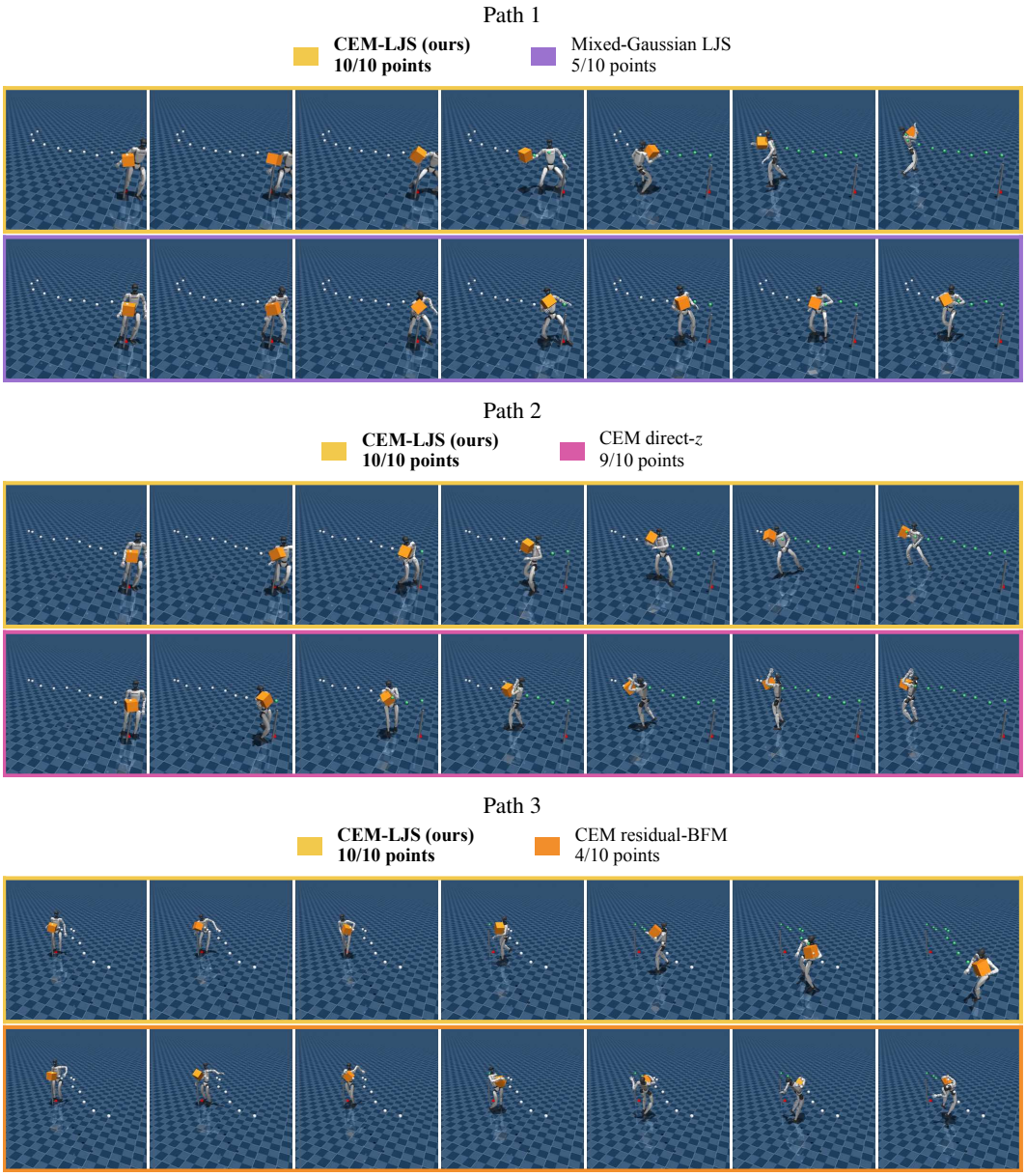


Figure 9: **Sanity check with minimal definitions.** CEM-LJS and one baseline on each path with the same seed, seven evenly spaced frames per run. The small spheres mark the points of the path and turn green once the box reaches them.

We use only the pretrained play policy, without any fine-tuning, and evaluate it in MuJoCo on the four assembly tasks of Play2Perfect. The effects are: pick up, mate, and release for the first beam

part, pick up, approach, mate, and release for the second beam part, pick up, engage, turn, and release for screwing, and pick up, seat, and release for the tight insertion. Screwing uses a different policy mode of the same checkpoint.

Search. CEM-LJS proposes changes to the 29 joint targets and maps them to the 6-dimensional goal through the damped inverse of the policy Jacobian. CEM direct-goal searches the six translation and rotation coordinates of the goal directly, with initial standard deviations of 0.1 m and 0.5 rad. Both use the same physical-effect scores, goal bounds, and elite selection, 1,024 candidates per decision (4,096 for the first beam part), a 20-step prediction, and a 10-step execution. The ten seeds were not used while defining the effects. Over the four tasks, CEM-LJS completes 31 of 40 runs against 21 for CEM direct-goal (Fisher's exact test, $p = 0.034$), and the largest gap is on screwing (9 against 3, $p = 0.020$). Table 17 gives the completed tasks with their search times, Figure 6 the search cost to reach each effect, Figure 10 frames from the runs, and Table 18 the individual runs.

Table 17: **Assembly with the frozen Play2Perfect policy, ten seeds.** Fully completed tasks, and the wall-clock search time of successful runs, mean (fastest), on an NVIDIA RTX 4090.

Task	CEM-LJS	CEM direct-goal	Time, CEM-LJS	Time, CEM direct-goal
Beam, first part	**10/10**	**10/10**	62 s (29 s)	46 s (29 s)
Beam, second part	**7/10**	4/10	72 s (43 s)	65 s (48 s)
Screwing	**9/10**	3/10	93 s (52 s)	132 s (88 s)
Insertion	**5/10**	4/10	83 s (69 s)	64 s (22 s)
All	**31/40**	21/40		

Table 18: **Wall-clock search time and outcome of every Play2Perfect evaluation run, ten seeds.** Each entry gives the accepted effects / task total and, in parentheses, the search time in seconds on an NVIDIA RTX 4090. Fully completed tasks are in bold.

Task	Method	929560953	875095668	1769421694	1418357545	885480420	1284918341	747747906	2023936781	1602613270	1370732405
Beam, first	CEM-LJS	**3/3 (106)**	**3/3 (36)**	**3/3 (47)**	**3/3 (46)**	**3/3 (35)**	**3/3 (29)**	**3/3 (66)**	**3/3 (93)**	**3/3 (62)**	**3/3 (99)**
	CEM direct-goal	**3/3 (29)**	**3/3 (76)**	**3/3 (41)**	**3/3 (96)**	**3/3 (42)**	**3/3 (51)**	**3/3 (30)**	**3/3 (34)**	**3/3 (33)**	**3/3 (33)**
Beam, second	CEM-LJS	1/4 (107)	3/4 (114)	1/4 (98)	**4/4 (60)**	**4/4 (75)**	**4/4 (56)**	**4/4 (119)**	**4/4 (51)**	**4/4 (96)**	**4/4 (43)**
	CEM direct-goal	2/4 (143)	1/4 (142)	1/4 (92)	1/4 (86)	1/4 (102)	**4/4 (51)**	**4/4 (48)**	1/4 (130)	**4/4 (72)**	**4/4 (91)**
Screwing	CEM-LJS	**4/4 (52)**	**4/4 (110)**	**4/4 (93)**	**4/4 (140)**	**4/4 (83)**	**4/4 (76)**	**4/4 (162)**	**4/4 (64)**	**4/4 (58)**	2/4 (156)
	CEM direct-goal	**4/4 (88)**	**4/4 (155)**	1/4 (125)	1/4 (147)	1/4 (121)	2/4 (152)	**4/4 (152)**	0/4 (114)	0/4 (123)	1/4 (150)
Insertion	CEM-LJS	**3/3 (106)**	**3/3 (76)**	**3/3 (78)**	**3/3 (88)**	1/3 (116)	1/3 (132)	1/3 (126)	1/3 (136)	**3/3 (69)**	1/3 (110)
	CEM direct-goal	1/3 (112)	1/3 (125)	**3/3 (35)**	1/3 (128)	**3/3 (95)**	1/3 (127)	**3/3 (103)**	1/3 (124)	**3/3 (22)**	1/3 (129)

I OTHER CONTROLLERS AND SIMULATORS

LJS needs only a frozen policy that takes a latent or a goal, and a simulator to evaluate candidates. To show how easily it carries over to any steerable latent- or goal-conditioned pretrained controller, we also steer the following controllers, embodiments, and simulators. Figure 11 shows frames from these searches, and their videos are available on the project page.

ASE (Peng et al., 2022), in Isaac Gym. With the frozen sword-and-shield controller, LJS opens a passive door, jumps across a gap and lands stably, and in a second task opens a swing gate through contact, walks through it, and stops upright.

FB-MEBE (Hu et al., 2026), in Isaac Sim. With the frozen policy of a Unitree Go2 quadruped, LJS pushes a passive box to a target and stops it there.

SPiRL (Pertsch et al., 2020), in the MuJoCo kitchen. Through the frozen skill decoder, LJS opens a cabinet, grasps a cube and pulls it out, carries it under the door, and pushes the door closed.

BFM-Zero in a second simulator. In Isaac Sim with PhysX, LJS picks up a breadbox, carries it, and places it, replanning in a separate simulator process while the robot keeps moving.

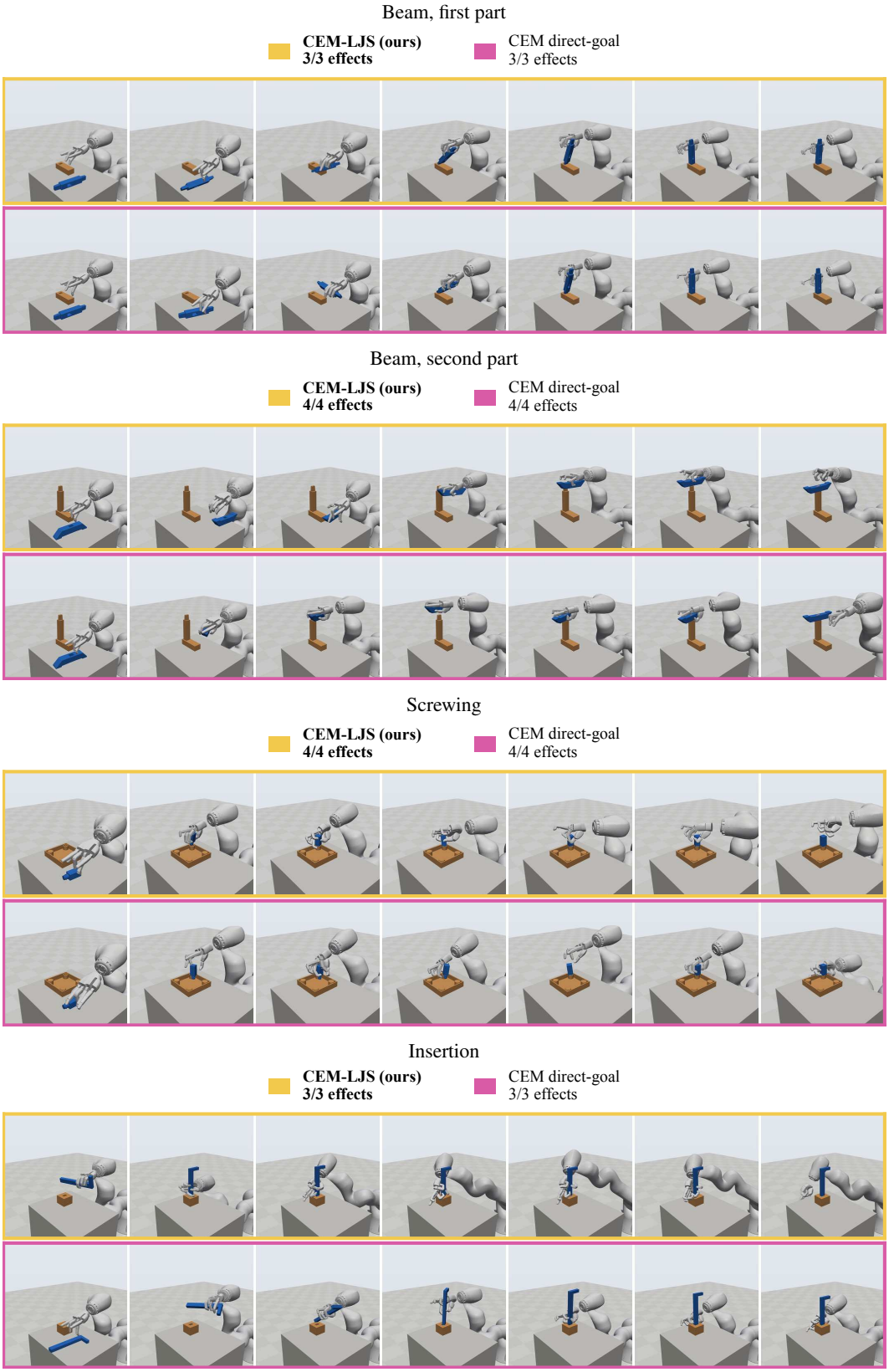


Figure 10: **Assembly with the frozen Play2Perfect policy.** Seven evenly spaced frames from a successful run of each method on each task.

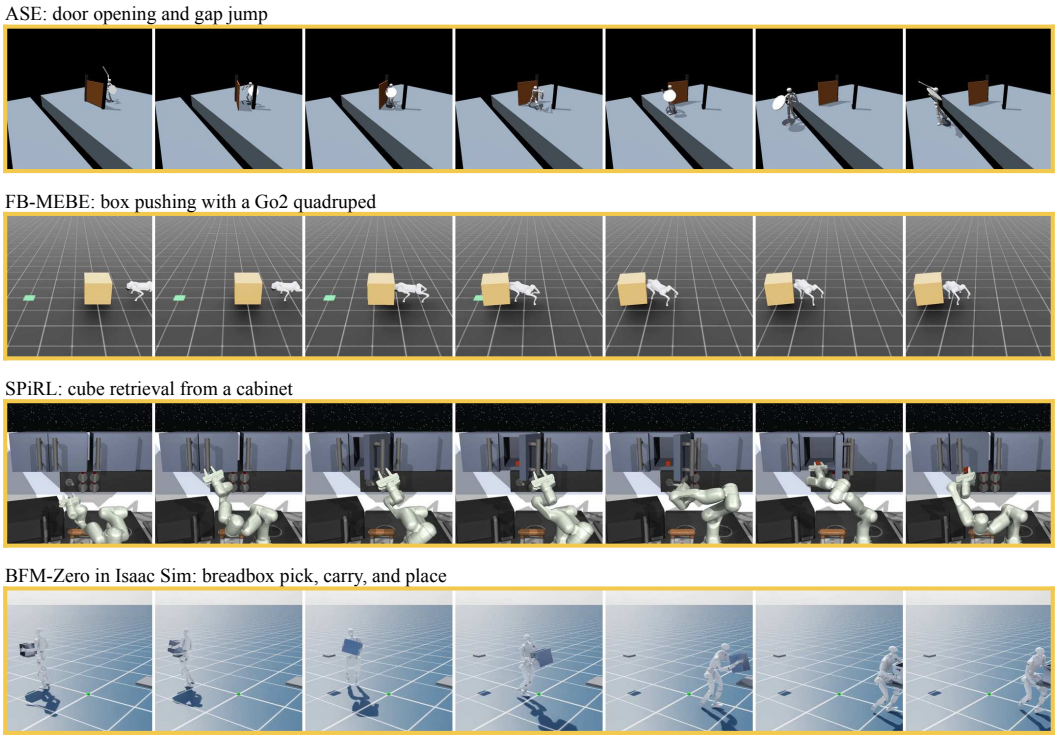


Figure 11: **LJS with further pretrained controllers and a second simulator.** Seven evenly spaced frames from one search per controller.

J QUALITATIVE ROLLOUT COMPARISONS

Figures 12 and 13 compare CEM-LJS with the best run of each baseline over the ten seeds. On door-cube, CEM-LJS and CEM direct-z both open the door with the grasped cube, while CEM residual-BFM lifts the cube but never brings it to the door. On the hurdle, the best run of every method completes the crossing, and Appendix E compares how they move. The videos of all evaluation runs are available on the project page.

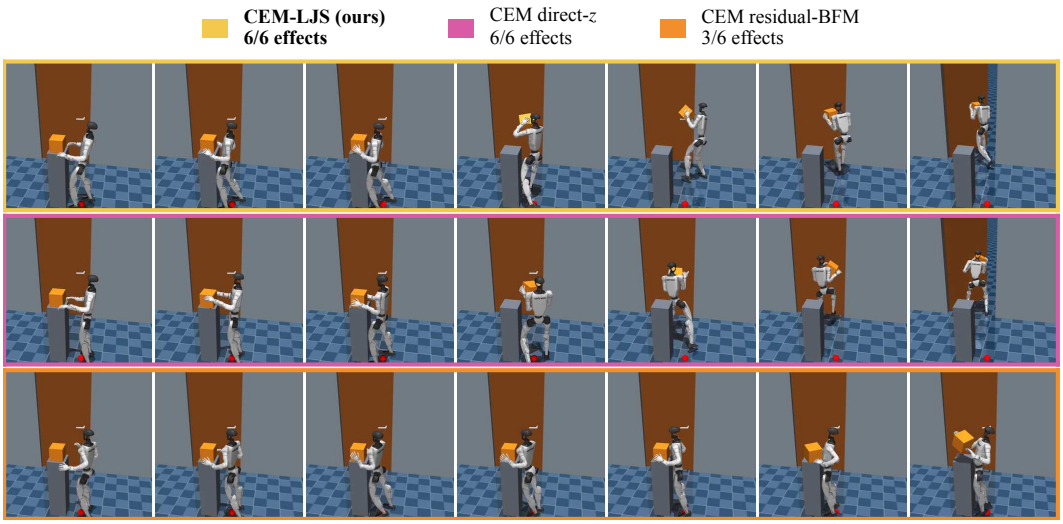


Figure 12: **Door opening while carrying a cube.** Seven evenly spaced frames from the best run of each method over the ten seeds. CEM-LJS and CEM direct-z complete all six effects, and CEM residual-BFM completes three.

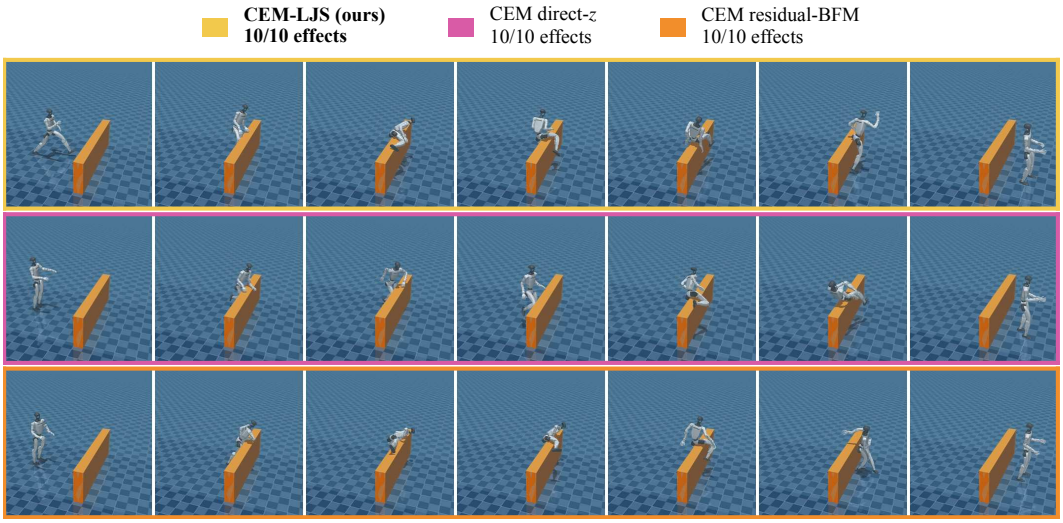


Figure 13: **Hurdle traversal.** Seven evenly spaced frames from the best run of each method over the ten seeds. Each of these runs completes all ten effects.

K PHYSICAL-EFFECT DEFINITIONS AND HUMAN–AI SCORE DEVELOPMENT

This appendix reports how the physical-effect definitions were developed and provides the mathematical scores and acceptance conditions for the three evaluated tasks. We would like to reiterate that our main contribution is LJS itself which makes the search for a specific physical effect productive. The following score functions are not part of our proposed LJS algorithm. They are to be viewed as evaluations of the kind of exploration that is possible with LJS, while impossible or highly sample-inefficient with other baselines. We leave the autonomous and efficient discovery of these physical effects for a given task in the broader sense to future work. The interaction card below follows the compact alternating-contribution format of Feng et al. (2026). The human specified the intended physical outcome, supplied the simulator conventions, corrected geometric and temporal assumptions, and accepted the final definition. The language model translated those instructions into

candidate measurement and score code. It did not observe rollouts, choose candidates, control the robot, or modify an effect during a reported search.

K.1 HUMAN–AI INTERACTION CARD: FIRST HURDLE EFFECT

Table 19: **Human–AI interaction card for the first hurdle effect.** The card records the substantive revisions rather than reproducing the full prompts and code responses. The final scores and acceptance conditions are given in Appendix K.2.3.

Contributor	Contribution
Human	Established the coding protocol: define one physical effect at a time for LJS with frozen BFM-Zero; implement measurements, progress scores, and acceptance conditions using the supplied MuJoCo interface; use simple scaled residuals; keep score and acceptance separate; and do not introduce joint-pose commands, inverse kinematics, motion tracking, fine-tuning, teleportation, or external forces. Any revised score must be rerun from the preceding accepted effect.
Gemini 3.8 Flash	Restated the protocol and requested the task geometry, entity identifiers, and first-stage instruction before generating code.
Human	Specified the fixed hurdle, crossing and lateral axes, right-leg lead, palm-normal conventions, collision-surface geometry, and contact-force convention. Requested the first effect: place both palms on the hurdle top, facing downward, with separate bilateral support, while keeping the robot within the usable width. Either hand could settle anywhere suitable; no exact hand target or extra foot-lifting stage was allowed.
Gemini 3.8 Flash	Proposed a complete first-stage module using collision geometry for palm placement, palm-normal alignment, per-hand contact loads, and pelvis centering, together with initial model-chosen tolerances.
Human	Corrected three implementation details: use each transformed hand geometry's bounding-box center for horizontal placement and its minimum vertex for height; measure support only at the final frame rather than claiming a sustained five-frame hold; and evaluate all surface bounds in the hurdle's live local frame instead of hardcoding its world center.
Gemini 3.8 Flash	Revised the measurements to use those distinct geometric statistics, instantaneous final-frame forces, and the hurdle transform recorded in each frame.
Human	Refined the intended outcome after inspecting the definition: keep both hands near the middle to leave room for the next leg movement, but avoid exact positions; require only reasonably downward palms and light bilateral load; reject side-friction support; keep the robot roughly centered; and stop penalizing small placement errors once the palms are suitably on top. Valid near-edge top contact had to count, so top support was to be identified from contact position and normal direction rather than an arbitrary edge strip.
Gemini 3.8 Flash	Added placement and orientation deadbands, a centered admissible band, light per-hand support targets, and top-face contact filtering using both local contact position and normal alignment. The human reviewed this proposal before the definition was incorporated into the finalized hurdle module.

Raw prompts and outputs: https://share.gemini.google/GgqoaJR2Yw4F

The card documents one representative effect rather than presenting the coding agent as an autonomous task designer. The successive corrections are part of the task-specification process: they remove assumptions that were inconsistent with the simulator semantics or with the intended physical outcome. The final reported searches use fixed, human-reviewed definitions.

K.2 MATHEMATICAL PHYSICAL-EFFECT DEFINITIONS

We give the scores and acceptance conditions for the ten hurdle effects, five shutter effects, and six door-cube effects, and the ten effects of the sanity check. Distances, forces, and times are in metres,

newtons, and seconds. Angles are in radians unless degrees are shown. Measurements are taken at the final frame of the effect history ξ, except where a temporal minimum or maximum is specified.

Residual and acceptance conventions. Define elementwise residuals

$$L_a(v;b) = \frac{[b-v]_+}{a}, \qquad U_a(v;b) = \frac{[v-b]_+}{a}, \qquad N_{a,d}(v;b) = \frac{[|v-b|-d]_+}{a}.$$

Vector scales and tolerances are applied componentwise. For each effect we list residual blocks $r_{k,j}$; its score is

$$c_k(\xi) = \sum_j \|r_{k,j}(\xi)\|_2^2.$$

By default, acceptance requires every residual block to vanish. A superscript g marks a guidance-only block excluded from acceptance. Explicit acceptance overrides below replace the zero-residual test for the named block. Requirements marked c additionally reject a candidate when their acceptance test fails on the full predicted effect segment. The scores are evaluated over successive trajectory prefixes and averaged over the latter half of the prediction horizon, as in the main method.

K.2.1 SHUTTER TRAVERSAL

Let $\mathcal{P}_G(s)$ be the transformed collision vertices of body group G, with $G = U$ for the head and torso and $G = R$ for the entire robot. The shutter's lower edge and far face are $y_b(s)$ and $x_f(s)$. Define

$$d_G(s) = \min_{p \in \mathcal{P}_G(s)} p_x - x_f(s), \qquad a_U(s) = \left[\max_{\substack{p \in \mathcal{P}_U(s) \\ p_x \le x_f(s)+.02}} p_z + .02 - y_b(s) \right]_+,$$

where $a_U = 0$ if the restricted vertex set is empty. For each hand, its placement point uses the mean vertex coordinates in x, y and the maximum vertex height. Let $e_h \in \mathbb{R}^6$ stack its displacement from the two targets $(x_{\text{panel}}, y_{\text{panel}} \pm .20, y_b - .005)$. Let $n_h \in \mathbb{R}^2$ contain the upward components of the palm normals. Finally, F_a is the signed net upward force on the panel from hand, wrist, and elbow contacts.

1. **Palms beneath the shutter.**

   $$r_1 = \operatorname{col}\big(N_{.10,.04}(e_h;0),\ L_1(n_h;\cos 30^\circ)\big).$$

   Acceptance instead permits $\|e_h\|_\infty \le .10$ and both upward components at least $\cos 45^\circ$.

2. **Lift the shutter.**

   $$r_2 = \operatorname{col}\big(L_{.10}(y_b;.75),\ L_{10}(F_a;1)\big).$$

   Acceptance requires a lower-edge height of at least .75 and at least 1 N of upward arm contact.

3. **Head and torso through.**

   $$r_3 = \operatorname{col}\big(U_{.10}(a_U;0),\ L_{.10}(d_U;.02)\big).$$

4. **Whole body clear.**

   $$r_4 = L_{.10}(d_R;.02).$$

5. **Stand on the far side.** Let $p^\star = (1.8, 0, z_{\text{nom}})$, where the horizontal point $(1.8, 0)$ lies .565 m beyond the far face on the centre line of the opening and z_{nom} is the root height of the nominal standing pose. Let e_{rot} be the quaternion rotation difference from the identity orientation, which faces $+x$ through the opening. With nominal joint positions q_{nom} and root speed $v = \|\dot{p}_{\text{root}}\|_2$, define

   $$e_{\text{stand}} = \operatorname{col}\big((p_{\text{root}} - p^\star)/.10,\ e_{\text{rot}}/.5,\ q_{\text{joints}} - q_{\text{nom}}\big), \qquad r_5 = \operatorname{col}\big(e^{\text{g}}_{\text{stand}},\ U_1(v;.10)\big).$$

   Acceptance requires at least .5 s of history and the following over the final .5 s: $\min_s (R_{\text{torso}}(s))_{zz} \ge \cos 10^\circ$, $\min_s d_R(s) \ge .02$, $\min_s z_{\text{root}}(s) \ge .9\, z_{\text{nom}}$, $\max_s v(s) \le .20$, and a floor normal load of at least 1 N on each foot in at least 80% of the frames.

K.2.2 DOOR OPENING WHILE CARRYING A CUBE

Let p_c, R_c, a denote the cube centre, rotation, and half-extents. The palm contact point is the collision-mesh mean shifted to the inward-most palm surface along the hand's local inward axis. For sides $\epsilon \in \{+1, -1\}$, define stacked palm errors relative to cube-local targets

$$e_{\mathrm{reach},\epsilon} = p_{h,\epsilon} - p_c - R_c(a_x + .03, \epsilon(a_y + .02), 0)^\top, \qquad e_{\mathrm{grasp},\epsilon} = p_{h,\epsilon} - p_c - R_c(.045, \epsilon a_y, 0)^\top.$$

Let $v_f \in \mathbb{R}^2$ be the dot products between normalized elbow-to-palm directions and the cube's forward axis. Let $b = (R_c^\top(p_c - p_{\mathrm{torso}}))_{xy}$, $u = (R_{\mathrm{torso}})_{zz}$, and v be the maximum root linear speed over the last .10 s. The maximum cube translation and rotation relative to its pose at the beginning of the effect are

$$m_p = \max_{s\in\xi} \|p_c(s) - p_c(s_0)\|_2, \qquad m_R = \max_{s\in\xi} \arccos\left(\frac{\mathrm{tr}(R_c(s_0)^\top R_c(s)) - 1}{2}\right),$$

with the arccos argument clipped to $[-1, 1]$. Define

$$r_m = U_{(.06,15^\circ)}\big((m_p, m_R); (.02, 5^\circ)\big), \qquad h_c = p_{c,z} - \sum_j |R_{c,zj}| a_j, \qquad h_s = h_c - z_{\mathrm{stand}}.$$

Here z_{stand} is the stand's top height. F_s is the sum of positive cube–stand normal loads. A bar below a measurement denotes its minimum over the final .08 s; a bar above denotes its maximum over that window.

Contact and separation measurements. $F_h \in \mathbb{R}^2$ contains the minimum inward palm loads over .08 s. For hand ϵ, a contact contributes when its cube-local position q satisfies $|q_y - \epsilon a_y| \le .02$, $|q_x| \le .075$, and $|q_z| \le .075$. Its contribution is $[-\epsilon(R_c^\top f_c)_y]_+$, where f_c is force on the cube. F_h^{full} instead permits $|q_x| \le a_x + .01$ and $|q_z| \le a_z + .01$. These load measurements return zero if the history is shorter than .08 s.

Let p_H, e_H, l_H, r_H denote the handle centre, axial direction, half-length, and radius. Its sampled surface gap to the cube is

$$g_H = \left[\min_{t\in\{-1,-.5,0,.5,1\}} \Big\| \big[\big|R_c^\top(p_H + t l_H e_H - p_c)\big| - a\big]_+ \Big\|_2 - r_H\right]_+.$$

For door centre p_D, normal n_D, and half-thickness a_D, define

$$g_D = \left[|(p_c - p_D)^\top n_D| - \sum_j |(R_c^\top n_D)_j| a_j - a_D\right]_+.$$

$F_H^{\downarrow}$ is the minimum summed downward cube force on the handle over .08 s; F_D is the minimum summed positive cube–panel normal load over the same window. Both return zero for shorter histories. The handle and door angles are θ_H and θ_D.

1. **Approach the cube.**

   $$r_1 = \mathrm{col}\big(N_{.20,.04}(e_{\mathrm{reach}}; 0),\ L_{.4}(v_f; \cos 30^\circ),\ N_{(.18,.20),(.04,.10)}(b; (.25, 0)),\ r_m^{\mathrm{c}},\ U_{.5}(v; .25),\ L_{.3}(u; \cos 25^\circ)\big).$$

   The motion block accepts $m_p \le .03$ and $m_R \le 8^\circ$; these are also candidate constraints.

2. **Grasp opposing side faces.**

   $$r_2 = \mathrm{col}\big(N_{.15,.02}(e_{\mathrm{grasp}}; 0),\ L_5(F_h; 5),\ r_m^{\mathrm{c}},\ U_{.5}(v; .25)\big).$$

   Each palm need only sustain 3 N for acceptance. The motion override is unchanged.

3. **Lift clear of the stand.**

   $$r_3 = \mathrm{col}\big(L_{.12}(h_s; .08),\ L_5(F_h; 5),\ U_5(F_s; 0),\ L_{.3}(u; \cos 30^\circ)\big).$$

   Acceptance permits $F_s \le .1$ and requires each palm load to be at least 3 N.

4. **Carry to the handle.**

$$r_4 = \mathrm{col}\big(U_{.20}(g_H; .02),\ (\sqrt{.5}\, N_{.20,.04}(e_{\mathrm{grasp}}; 0))^{\mathrm{g}},\\ L_{.20}(\underline{h}_c; .55)^{\mathrm{c}},\ L_5(F_h; 5),\ U_5(\overline{F}_s; 0)^{\mathrm{c}},\ L_{.3}(u; \cos 30^\circ)\big).$$

The palm-load override remains 3 N per hand; the stand-load override and constraint permit $\overline{F}_s \le .1$.

5. **Depress the handle with the cube.**

$$r_5 = \mathrm{col}\big(L_{.35}(\theta_H; .50),\ L_8(F_H^{\downarrow}; 8),\ U_{.10}(g_H; 0),\\ U_{.15}(\theta_D; .04),\ L_{.20}(h_c; .15),\ L_5(F_h; 5)^{\mathrm{c}},\ L_{.3}(u; \cos 30^\circ)\big).$$

Acceptance permits $\theta_D \le .06$ and requires $F_H^{\downarrow} \ge 3$ and both palm loads at least 3 N. The palm-load condition is also a candidate constraint.

6. **Push the door open.**

$$r_6 = \mathrm{col}\big(L_{.5}(\theta_D; 1),\ L_5(F_D; 5),\ U_{.10}(g_D; 0),\\ L_{.20}(h_c; .15),\ L_5(F_h^{\mathrm{full}}; 5),\ L_{.3}(u; \cos 30^\circ)\big).$$

Acceptance requires $\theta_D \ge .95$, $F_D \ge 3$, and each full-face palm load at least 3 N.

K.2.3 HURDLE TRAVERSAL

The fixed hurdle has near and far faces x_n, x_f, top height z_h, and half-width w. The right leg leads and the left follows. For collision group G, let x_G, z_G be the minimum world x, z coordinates of its collision surface. Mesh vertices and sphere extrema provide these measurements. Write T_G and F_G for its upward hurdle-top load and positive floor normal load. A top contact is counted only if its height is at least $z_h - .015$ and its contact normal has vertical magnitude greater than .7. Let A_n be the summed positive normal load on hands, wrists, and elbows at contacts with $x \le x_n + .015$ and $z < z_h - .015$. Define

$$C_b = U_{.20}(|y_{\mathrm{root}}|; b), \qquad K = U_{20}(A_n; 2), \qquad B(\xi) = \left[\cos 35^\circ - \min_{s\in\xi}(R_{\mathrm{torso}}(s))_{zz}\right]_+ .$$

K is always guidance-only. Let $u = (R_{\mathrm{torso}})_{zz}$, $l = (R_{\mathrm{torso}})_{xz}$, $f_y = (R_{\mathrm{torso}})_{yx}$, and W be total model mass times gravitational acceleration. Subscripts R, L denote right and left feet; $\mathrm{leg}R, \mathrm{leg}L$ include collision bodies descending from the respective knee. $A, P, \mathcal{R}$ denote arms, pelvis, and entire robot.

Placement and clearance measurements. For each palm's collision bounding-box centre b_h, define

$$e_h = \mathrm{col}\left(b_{h,x} - \mathrm{clip}(b_{h,x}, x_n + .025, x_f - .025),\ b_{h,y} - \mathrm{clip}(b_{h,y}, -w_p, w_p),\ z_h^{\mathrm{hand}} - z_h\right),$$

where $w_p = \min(.35, w - .08)$ and z_h^{hand} is the hand's lowest collision point. Stack both hands into e_H; n_H stacks downward palm-normal components. Let g_G be the minimum vertex distance of group G to the top rectangle $[x_n + .02, x_f - .02] \times [-w + .08, w - .08] \times \{z_h\}$. The pelvis error e_P comprises its bounding-box centre's signed distance from $[x_n + .02, x_f - .02]$ in x and its lowest collision height minus z_h.

Define right-foot and trailing-leg clearance errors

$$a_R = \begin{cases} [z_h + .02 - z_R]_+, & x_R < x_f + .02, \\ 0, & \text{otherwise}, \end{cases} \qquad a_L = \left[z_h + .02 - \min_{\substack{p\in\mathcal{P}_{\mathrm{leg}L} \\ p_x < x_f + .02}} p_z\right]_+ ,$$

with $a_L = 0$ when no vertices remain. Let $d_L = [.05 - \min(d_{\mathrm{knee}}, d_{\mathrm{ankle}})]_+$, where each d is the left joint's displacement from the pelvis projected onto the torso's forward axis. Finally, $v = \|\dot{p}_{\mathrm{root}}\|_2$ and $\omega = \|\omega_{\mathrm{root}}\|_2$.

1. **Both palms on top.**
$$r_1 = \mathrm{col}\big(C_{.36}, U_{.20}(|e_H|; .035), L_1(n_H; \cos 40^\circ), L_2(T_{H_L}; 2), L_2(T_{H_R}; 2)\big).$$
2. **Right leg on top for support.**
$$r_2 = \mathrm{col}\big(C_{.36}, U_{.20}(g_{\mathrm{leg}R}; 0), L_{.20}((p_{\mathrm{knee}R,x}, p_{\mathrm{knee}R,z}); (x_n + .04, z_h + .08))^{\mathrm{g}},$$
$$L_{10}(T_{\mathrm{leg}R}; 10), L_{20}(\min_{s\in\xi} T_A(s); 5)^{\mathrm{c}}, L_{.20W}(T_A; .20W)^{\mathrm{g}}, K^{\mathrm{g}}\big).$$
The leg-gap block accepts $g_{\mathrm{leg}R} \le .025$.
3. **Right foot on top.**
$$r_3 = \mathrm{col}\big(L_{.20}(x_R; x_n + .02), L_{.20}(z_R; z_h - .01), L_5(T_R; 5), L_{20}(F_L; 5), K^{\mathrm{g}}\big).$$
4. **Lean and sweep the right foot across.**
$$r_4 = \mathrm{col}\big(C_{.36}, L_{.30}(l; \sin 30^\circ), U_{.20}(a_R; 0), L_{.20}(x_R; x_f{+}.02), L_{20}(F_L; 5), L_{10}(T_{\mathcal{R}}; 10), K^{\mathrm{g}}\big).$$
5. **Lift the left foot into a supported straddle.**
$$r_5 = \mathrm{col}\big(C_{.35}, L_{.30}(l; \sin 10^\circ), L_{.20}(x_R; x_f + .02), L_{.20}(z_L; .08),$$
$$U_W(F_L; 0)^{\mathrm{g}}, L_{.20}(x_{\mathrm{root}}; x_n + .02), L_{20}(T_{\mathcal{R}}; 10), K^{\mathrm{g}}\big).$$
6. **Settle into an upright supported straddle.**
$$r_6 = \mathrm{col}\big(C_{.36}, L_{.30}(u; \cos 15^\circ), L_{.30}(f_y; \cos 30^\circ), e_P/.20,$$
$$L_{.20}(x_R; x_f + .02), U_{.30}(p_{\mathrm{ankle}R,z} - p_{P,z}; -.10), L_{.20}(z_L; .08),$$
$$U_{.50}(v; .20), U_1(\omega; .50), L_{10}(T_P; 10), K^{\mathrm{g}}\big).$$
The pelvis-placement block accepts $\|e_P\|_\infty \le .025$.
7. **Left foot on the ledge.**
$$r_7 = \mathrm{col}\big(g_L/.20, d_L/.20, L_{20}(T_L; 10), L_{.20}(z_L; z_h - .01)^{\mathrm{g}}, B(\xi)/.20\big).$$
Acceptance requires $g_L/.20 \le .025$, $d_L = 0$, $T_L \ge 10$, and terminal $u \ge \cos 35^\circ$. The historical balance penalty may remain nonzero.
8. **Left leg clears the far edge.**
$$r_8 = \mathrm{col}\big(L_{.20}(\min_{s\in\xi} x_L(s); x_n - .04)^{\mathrm{c}}, L_{.20}(x_L; x_f + .03), a_L/.20, B(\xi)/.20\big).$$
The forward residual accepts $L_{.20}(x_L; x_f + .03) \le .05$; the remaining blocks use their default zero tests.
9. **Both feet on the far-side floor.**
$$r_9 = \mathrm{col}\big(L_{.20}(\min_{s\in\xi} x_L(s); x_f + .01)^{\mathrm{c}},$$
$$\{L_{.20}(x_i; x_f + .03), U_{.20}(z_i; .04), L_{10}(F_i; 5)\}_{i\in\{L,R\}}, (B(\xi)/.20)^{\mathrm{g}}\big).$$
10. **Stand and settle.** Let $p^\star = (x_f + .50, y_{\mathrm{root}}, z_{\mathrm{nom}})$ and let e_{rot} be the quaternion rotation difference from the identity orientation. With nominal joint positions q_{nom}, define
$e_{\mathrm{stand}} = \mathrm{col}\big((p_{\mathrm{root}} - p^\star)/.20, e_{\mathrm{rot}}/.5, q_{\mathrm{joints}} - q_{\mathrm{nom}}\big)$, $\quad r_{10} = \mathrm{col}\big(e^{\mathrm{g}}_{\mathrm{stand}}, [\overline{v} - .15]_+\big)$.
Here temporal extrema use the final .5 s. Acceptance requires at least .5 s of history, $\underline{u} \ge \cos 15^\circ$, $\overline{v} \le .15$, $\min_{i\in\{L,R\}} \underline{F}_i \ge 5$, and $\min_s(x_{\mathcal{R}}(s) - x_f) \ge .03$ over that window.

K.2.4 Carrying a box through points

This is the task of the sanity check in Appendix G. Let p_b be the box centre and q_k the k-th point of the path, in a world frame with x forward, y to the robot's left, and z up. The box starts on the post at $(.306, 0, .891)$. With t_{end} the time of the final frame of ξ, the time since the box first touched the floor is
$$\tau_{\mathrm{floor}}(\xi) = \Big[t_{\mathrm{end}} - \min\{t(s) : s \in \xi, \text{ the box touches the floor at } s\}\Big]_+,$$
with $\tau_{\mathrm{floor}} = 0$ if the box never touches the floor. Every effect $k = 1, \dots, 10$ has the same two blocks,
$$r_k = \mathrm{col}\big((p_b - q_k)/.10,\ \tau^{\mathrm{c}}_{\mathrm{floor}}\big).$$
The point block accepts $\|p_b - q_k\|_2 \le .05$, and the floor block requires $\tau_{\mathrm{floor}} = 0$. Nothing else is measured. Table 20 lists the points of the three paths.

Table 20: **Points of the three generated paths** for the sanity check, box-centre positions in metres.

Point	Path 1	Path 2	Path 3
1	(0.300, −0.050, 0.980)	(0.300, −0.050, 0.980)	(0.300, 0.050, 0.980)
2	(0.303, −0.350, 0.985)	(0.425, −0.323, 0.940)	(0.403, 0.332, 0.963)
3	(0.458, −0.607, 0.940)	(0.462, −0.621, 1.007)	(0.623, 0.535, 1.001)
4	(0.408, −0.903, 0.940)	(0.351, −0.899, 0.957)	(0.665, 0.832, 1.068)
5	(0.488, −1.192, 0.940)	(0.257, −1.184, 0.986)	(0.901, 1.017, 1.129)
6	(0.464, −1.491, 0.940)	(0.318, −1.478, 1.032)	(1.182, 1.124, 1.073)
7	(0.561, −1.775, 1.014)	(0.446, −1.749, 1.018)	(1.355, 1.369, 1.067)
8	(0.554, −2.075, 1.058)	(0.549, −2.031, 1.069)	(1.586, 1.560, 0.987)
9	(0.338, −2.283, 1.102)	(0.477, −2.322, 1.124)	(1.747, 1.813, 0.940)
10	(0.042, −2.333, 1.090)	(0.308, −2.570, 1.058)	(1.803, 2.108, 0.940)

L TIPS FOR PRACTITIONERS

Begin with an intuitive score. Highly constrained scores are not necessarily better scores. We first encode the obvious physical outcome and inspect the behavior produced by the search. We add terms only to address failure modes observed in the resulting rollouts, rather than trying to anticipate every possible failure in advance.

Define the task one physical effect at a time. The terminal rollout of one accepted effect often reveals how the next effect should be specified. It may expose the relevant contact configuration, show which movement is currently blocked, or indicate that an intermediate physical effect is required. We therefore found it more practical to construct long tasks stage-wise than to define every effect in advance.

Inspect the motion, not only the scalar score. A lower physical-effect score does not always indicate useful progress, and a temporary score increase does not always indicate failure. The robot may need to make a short adjustment before further progress becomes possible, while a search can also exploit a missing condition and reduce the score through an unintended movement. Video replay remains the most useful diagnostic for distinguishing these cases.

Avoid ending a stage in uncontrolled motion. Dynamic or unexpected terminal states are difficult to recover from during later effects. When this occurs, the next score should not merely force the robot forward from an unsuitable state. It is usually better to revise the preceding effect to require settling, support, or bounded velocity before acceptance.

Match the horizon to the physical response. Large body movements and heavy-object displacement may not develop within the same horizon as a small contact adjustment. Increasing the prediction horizon gives the simulator time to reveal the outcome of a candidate, while increasing the execution horizon can reduce the number of replanning decisions. Both can also make an incorrect candidate more costly. We therefore treat prediction and execution horizons as task-stage parameters and report them for every effect.