

Generative Flow on Symbolic Abstractions for Embodiment Capability Discovery

Youwei Yu Lantao Liu
Indiana University Bloomington
{youwyu, lantao}@iu.edu

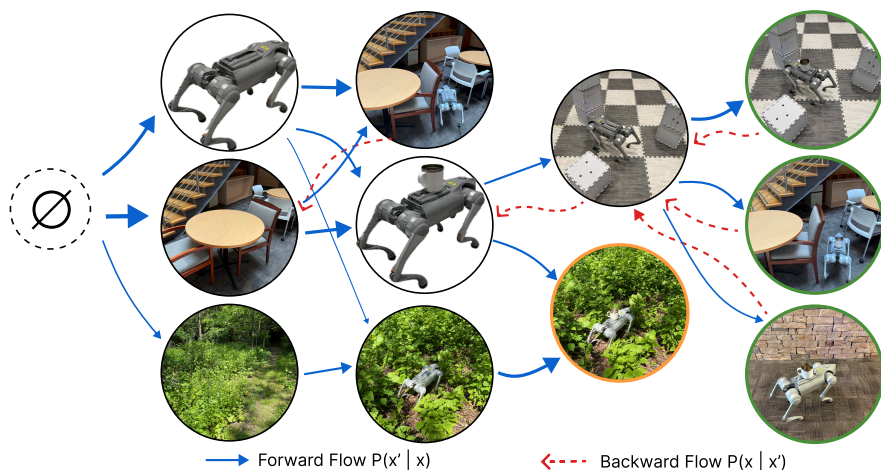


Figure 1: FOSA learns a generative flow policy over symbolic task skeletons to discover tasks near a robot’s empirical capability frontier. Starting from the empty skeleton $\emptyset$, the policy samples grammar-valid semantic edits with transition probability $P(x' | x)$. Blue arrow $\rightarrow$ thickness indicates sampling probability.

Abstract: Achieving complex, long-horizon robot behavior remains challenging because successful execution depends on skills grounded in a robot’s physical affordances. Existing training-task distributions are often handcrafted or sampled from large language models (LLMs), relying on semantic plausibility rather than embodiment-specific feasibility. This mismatch can produce curricula that are too difficult, redundant, or blind to non-obvious but learnable behaviors. We propose Generative Flow on Symbolic Abstractions (FOSA), a framework for discovering embodiment-specific capabilities by learning a grounded task generator. FOSA parameterizes an abstract policy as a Generative Flow Network that constructs symbolic task skeletons. Each skeleton is compiled into a simulation environment, where low-level policy-learning performance provides the reward signal for the abstract policy. Across heterogeneous robot platforms, FOSA discovers diverse and executable tasks near the robot’s empirical capability frontier. After distilling the discovered task policies into a single language-conditioned policy, FOSA improves performance on unseen complex and long-horizon tasks. The code and demonstration videos are available at <https://www.youwei-yu.com/fosa>.

Keywords: Robot foundation models, robotic data generation, sim-to-real transfer

1 Introduction

Robot learning has become increasingly effective once the target task is specified and lies within the robot’s learnable regime. With a well-specified reward and simulator, reinforcement learning (RL) with domain and dynamics randomization can produce agile policies that transfer from simulation to hardware [1, 2, 3, 4, 5]. With demonstrations, expressive policy classes can represent multimodal

behaviors in contact-rich manipulation [6]. These successes still presuppose that the task distribution has been chosen. In practice, robot embodiments are usually designed for broad classes of use rather than a single task, leaving their actual capability space only partially explored.

The choice of task distribution becomes critical for complex and long-horizon policy learning. Skill composition, hierarchical RL, and task-and-motion planning all rely on intermediate capabilities that can be invoked, chained, or refined [7, 8, 9, 10]. Yet these capabilities are often assumed known, manually designed, or proposed from human semantic intuition. These human-specified structures are brittle across embodiments: a task that appears simple may be physically difficult for a robot, while a non-obvious behavior may be easy to learn and useful for composition [11].

We study *embodiment capability discovery*: finding the task structures that a specific robot can learn under a fixed training budget. The central object is the robot’s empirical capability frontier, consisting of tasks that are learnable but not already trivial. This frontier is useful for two reasons. First, frontier tasks expose skills that can support interpolation, composition, and downstream long-horizon generalization. Second, the full task space is combinatorial, so exhaustive enumeration is infeasible; efficient discovery requires learning where useful capabilities are likely to emerge.

We propose Generative Flow on Symbolic Abstractions (FOSA), a framework that learns an abstract task generator over symbolic task skeletons. FOSA samples a skeleton, compiles it into a simulation environment, trains a low-level task policy, and uses the resulting performance to update the generator. We instantiate the generator as a Generative Flow Network [12], enabling diverse sampling over variable-length and variable-dimensional task specifications near the capability frontier. Language models are used as compilation assistants for converting symbolic skeletons into executable environments. Our contributions are summarized as follows:

- We propose FOSA, a general framework for embodiment capability discovery that learns an abstract policy over symbolic task skeletons and grounds task selection in low-level policy-learning performance.
- We instantiate the abstract policy with a Generative Flow Network, enabling diverse search over variable-length and variable-dimensional task specifications near the robot’s empirical capability frontier.
- We evaluate FOSA across heterogeneous robot platforms in simulation and the real world. We find that proximity to the discovered capability frontier correlates with generalization to held-out complex and long-horizon tasks, and that FOSA improves performance compared with task-generation and curriculum-learning baselines.

2 Related Work

Complex and Long-Horizon Robot Behavior. Complex robot policy often requires temporal abstraction, reusable skills, and composition over extended horizons. Hierarchical RL and options provide classical mechanisms for learning and invoking temporally extended actions [7, 8, 13], while skill-prior and skill-chaining methods improve long-horizon control by reusing learned behavioral primitives [14, 15, 16, 17]. Task-and-motion planning and bilevel formulations similarly decompose long-horizon problems into symbolic decisions and continuous control problems [9, 10]. In robotics, platform-specific learning has produced increasingly rich behaviors, including agile quadruped locomotion, dexterous manipulation, and articulated-object interaction [4, 18, 19, 20, 21]. These works demonstrate the importance of skills and compositional structure, while typically assuming that the relevant task family, skill library, or decomposition is already specified.

Skill Discovery and Open-Ended Curricula. A complementary line of work studies how agents can acquire skills, goals, or curricula with less manual supervision. Unsupervised skill discovery learns diverse behaviors through intrinsic objectives based on mutual information, state coverage, empowerment, contrastive representation learning, or predictable skill-conditioned dynamics [22, 23, 24, 25]. Goal-generation and curriculum-learning methods adapt the training distribution according to learning progress, regret, success rate, or task difficulty [26, 27, 28]. Open-ended

environment design further studies how to generate environments that induce increasingly complex agent behavior [29, 30, 31, 32, 33]. These methods reduce manual ordering of experience, but often operate within a fixed task family, fixed simulator parameterization, or motor-level behavior space. FOSA instead searches over symbolic task skeletons and scores each candidate through embodiment-specific policy learning.

Language-Driven Generation and Scalable Robot Learning. Large language models have recently been used to scale robot learning by generating tasks, rewards, curricula, code, simulation scenes, and supervision [34, 35, 36, 37, 38, 39, 40, 41, 42]. These systems reduce manual engineering and make it possible to instantiate more diverse robot-learning problems. In parallel, sim-to-real methods improve transfer through domain randomization, dynamics randomization, system identification, and online adaptation [1, 2, 3, 4, 5]. Recent robot foundation models and vision-language-action policies scale policy learning with large heterogeneous robot datasets [43, 44, 45, 46], while morphology-conditioned control and robot-design methods show that policy learning is strongly coupled to embodiment [47, 48, 49]. FOSA uses the LLM only to compile skeletons and propose edits; task retention is decided by embodiment-grounded policy learning.

3 Preliminaries

We formulate robot capability discovery as learning an abstract policy over symbolic task skeletons and grounding each sampled skeleton through low-level policy learning. We first define the task abstraction used by FOSA, then review the GFlowNet objective used to sample diverse skeletons in proportion to embodiment-grounded utility.

Symbolic Abstractions. We represent each candidate robot-learning problem as a symbolic task skeleton, following the spirit of task-and-motion planning (TAMP) [9, 10]. The symbolic level specifies objects, goals, and operators, while the continuous level specifies the geometric, dynamical, and simulator parameters required for execution. Let $\mathcal{Y}$ denote entity types, $\mathcal{P}$ predicate schemas, $\mathcal{U}$ operator schemas, and Φ feasibility constraints. For a finite set of typed entities E , let $\text{Atoms}(\mathcal{P}, E)$ and $\text{Lits}(\mathcal{P}, E)$ denote the induced ground atoms and literals. A task skeleton is defined as

$$\mathbf{x} = [E_{\mathbf{x}}, I_{\mathbf{x}}, G_{\mathbf{x}}, U_{\mathbf{x}}, \theta_{\mathbf{x}}, \Phi_{\mathbf{x}}], \quad (1)$$

where $E_{\mathbf{x}}$ is the set of typed task entities, $I_{\mathbf{x}} \subseteq \text{Atoms}(\mathcal{P}, E_{\mathbf{x}})$ is the initial symbolic state, $G_{\mathbf{x}} \subseteq \text{Lits}(\mathcal{P}, E_{\mathbf{x}})$ is the goal condition, $U_{\mathbf{x}} \subseteq \mathcal{U}$ is the set of referenced operator schemas, $\theta_{\mathbf{x}}$ stores continuous arguments, and $\Phi_{\mathbf{x}} \subseteq \Phi$ stores feasibility constraints such as valid poses, contact modes, tolerances, and parameter ranges. For example, in a quadruped slippery-slope locomotion task, $E_{\mathbf{x}}$ contains the robot and ramp, $G_{\mathbf{x}}$ encodes velocity tracking, $\theta_{\mathbf{x}}$ stores the incline and friction coefficient, and $\Phi_{\mathbf{x}}$ constrains feasible contacts and termination thresholds.

Given a robot embodiment $\mathbf{e}$, a complete and valid skeleton is compiled into an executable RL environment

$$\mathcal{M}_{\mathbf{e}, \mathbf{x}} = \text{Env}(\mathbf{e}, \mathbf{x}) = (\mathcal{O}_{\mathbf{e}, \mathbf{x}}, \mathcal{A}_{\mathbf{e}, \mathbf{x}}, p_{\mathbf{e}, \mathbf{x}}, r_{\mathbf{e}, \mathbf{x}}, \rho_{\mathbf{e}, \mathbf{x}}, \Gamma_{\mathbf{e}, \mathbf{x}}), \quad (2)$$

where the tuple denotes observation space, action space, transition dynamics, reward, reset distribution, and termination conditions. A skeleton is valid if it is type-consistent and supported by the embodiment and simulator. It is complete if all information needed for policy training is specified.

Capability Frontier. For a completed skeleton $\mathbf{x}$, FOSA trains a task policy $\pi_{\mathbf{x}}$ in $\mathcal{M}_{\mathbf{e}, \mathbf{x}}$ under a fixed sample budget and records its success rate before training, $\mathcal{J}_{\mathbf{x}}^0$, and after training, $\mathcal{J}_{\mathbf{x}}$, both in $[0, 1]$. We define the empirical capability frontier as the tasks whose success rate training improves but does not saturate,

$$\mathcal{B}_{\mathbf{e}} = \{\mathbf{x} : \mathcal{J}_{\mathbf{x}}^0 < \mathcal{J}_{\mathbf{x}} < 1\}. \quad (3)$$

Tasks on this frontier are learnable but nontrivial, and therefore provide informative feedback for capability discovery.

Generative Flow Networks. A Generative Flow Network (GFlowNet) [12] learns a stochastic construction policy whose terminal samples are distributed in proportion to a positive reward function.

In FOSA, the construction graph is a directed acyclic graph over partial skeletons. A trajectory

$$\tau = (\mathbf{x}^0 \rightarrow \mathbf{x}^1 \rightarrow \dots \rightarrow \mathbf{x}^T = \mathbf{x}), \quad \mathbf{x}^0 = \emptyset, \quad (4)$$

sequentially edits a partial skeleton until it reaches a complete skeleton. The forward policy $P_F(\mathbf{x}^{t+1} | \mathbf{x}^t)$ samples the next valid edit, and the backward policy $P_B(\mathbf{x}^t | \mathbf{x}^{t+1})$ scores valid predecessor states. Given terminal reward $R(\mathbf{x}) > 0$, the target distribution is $P^*(\mathbf{x}) \propto R(\mathbf{x})$. FOSA trains the abstract policy using trajectory balance [50]:

$$\mathcal{L}_{\text{GFlowNet}}(\tau) = (\log Z + \log P_F(\tau) - \log R(\mathbf{x}) - \log P_B(\tau))^2, \quad (5)$$

where $P_F(\tau) = \prod_{t=0}^{T-1} P_F(\mathbf{x}^{t+1} | \mathbf{x}^t)$, $P_B(\tau) = \prod_{t=0}^{T-1} P_B(\mathbf{x}^t | \mathbf{x}^{t+1})$, and Z is a learned scalar.

4 Method

FOSA discovers embodiment-specific capabilities by coupling an abstract policy over task skeletons with task policies optimized in generated environments. Given a robot embodiment $\mathbf{e}$, a symbolic domain $(\mathcal{Y}, \mathcal{P}, \mathcal{U}, \Phi)$, a compiler $\text{Env}(\cdot)$, and a fixed evaluation budget, FOSA returns an archive of discovered skeletons and trained policies. The archive is subsequently distilled into a general language-conditioned policy.

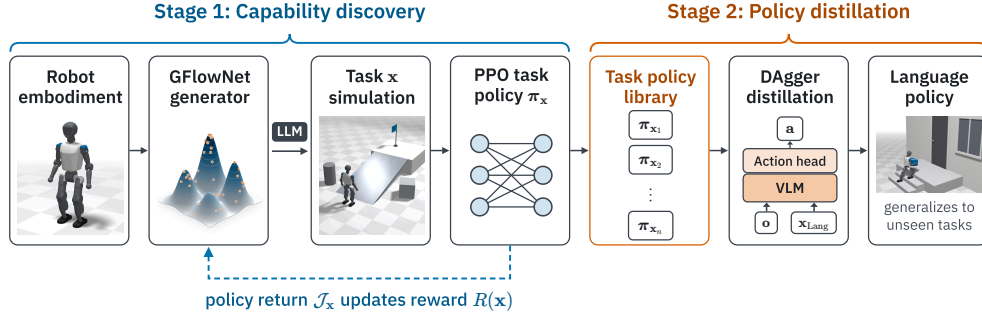


Figure 2: FOSA framework overview. The GFlowNet abstract policy generates symbolic task skeletons, which are compiled by an LLM into simulation environments to train task policies. Task-policy performance provides the reward signal that updates the abstract policy toward the robot’s empirical capability frontier.

4.1 FOSA Framework

At each discovery iteration, the abstract policy samples a construction trajectory from $\mathbf{x}^0 = \emptyset$ to a terminal skeleton $\mathbf{x}$. Deterministic validity and completeness checks reject skeletons that violate grammar, simulator, or embodiment constraints. The compiler instantiates $\mathcal{M}_{\mathbf{e}, \mathbf{x}} = \text{Env}(\mathbf{e}, \mathbf{x})$. A task policy $\pi_{\mathbf{x}}$ is trained from scratch in this environment using PPO [51]; the success rate of the first evaluation is $\mathcal{J}_{\mathbf{x}}^0$ and that of the last is $\mathcal{J}_{\mathbf{x}}$. FOSA converts them into $R(\mathbf{x})$, updates the GFlowNet via Eq. (5), and adds non-duplicate successful skeletons to the archive.

The LLM enters FOSA in two roles, neither of which selects tasks. Inside $\text{Env}(\cdot)$ it is a compiler: from the verbalized and serialized skeleton it writes a staged JSON specification (e.g., simulation contract and reward) [36, 37, 40]. During sampling, it also supplies a prior over valid edits. Which skeletons are retained and reinforced is decided solely by $R(\mathbf{x})$, so FOSA is grounded by physical policy-learning performance rather than by semantic plausibility.

4.2 FOSA Abstract Policy

The abstract policy must generate variable-dimensional task specifications through variable-length construction sequences, since different embodiments require different entities, constraints, goals, and arguments. FOSA parameterizes this policy as a GFlowNet and trains it with local-search sampling [52]. Every complete skeleton has exactly one goal atom, which defines its goal family $g(\mathbf{x})$; FOSA learns one $\log Z_g$ per family in Eq. (5), so that within each family terminal skeletons are sampled in proportion to $R(\mathbf{x})$ rather than collapsing onto a single best task.

Abstract State. An abstract state at construction step t is a partial skeleton

$$\mathbf{x}^t = [E_{\mathbf{x}^t}, I_{\mathbf{x}^t}, G_{\mathbf{x}^t}, U_{\mathbf{x}^t}, \boldsymbol{\theta}_{\mathbf{x}^t}, \Phi_{\mathbf{x}^t}]. \quad (6)$$

FOSA evaluates two deterministic predicates over this structure. The validity predicate $\text{Valid}_{\mathbf{e}, \mathcal{M}}(\mathbf{x}^t)$ checks type consistency, asset support, robot compatibility, and admissible parameter ranges. The completeness predicate $\text{Complete}_{\mathbf{e}, \mathcal{M}}(\mathbf{x}^t)$ checks whether all entities, goals, success predicates, reward terms, and continuous arguments required for environment compilation are specified. A skeleton is complete when it binds a goal, an entity or environment, and a constraint in at least four bindings with all required arguments set. A maximum horizon of $T_{\max}$ edits prevents redundant or circular task descriptions. P_F and P_B read an LLM embedding of the serialized partial skeleton, including its arguments.

Abstract Action. An abstract action $\mathbf{a}^t$ is a symbolic edit that maps $\mathbf{x}^t$ to $\mathbf{x}^{t+1}$. Invalid edits are masked before sampling from $P_F(\mathbf{x}^{t+1} \mid \mathbf{x}^t)$. The action space contains four edit classes: `add_atom`, which appends a typed atom (capability, entity or environment, goal, or constraint) with default arguments; `compose`, which adds a composition atom that binds existing components, once the skeleton has four bindings; `refine_arg`, which moves one numeric argument of $\boldsymbol{\theta}_{\mathbf{x}^t}$ by one of five evenly spaced levels between its bounds, or sets a categorical argument to another admissible value, once the skeleton is complete; and `stop`, which terminates the trajectory. The `stop` action is enabled only when $\text{Complete}_{\mathbf{e}, \mathcal{M}}(\mathbf{x}^t) = 1$.

Abstract Reward. The abstract reward favors skeletons on the empirical capability frontier that are structurally novel and concise. For a completed and trained skeleton $\mathbf{x}$, FOSA computes the utility

$$\mathcal{V}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \lambda_{\text{nov}} \mathcal{N}(\mathbf{x}) - \lambda_{\text{cost}} \mathcal{C}(\mathbf{x}), \quad (7)$$

where $\mathcal{N}(\mathbf{x}) = \max(0, \sum_{k \in \{\text{atom}, \text{pair}, \text{sig}\}} w_k \overline{(1+n)^{-1/2}} - d(\mathbf{x}))$ is a count-based novelty over the skeleton’s atoms, atom pairs, and action–entity–condition signatures; for each kind k the bar averages $(1+n)^{-1/2}$ over the skeleton’s components of that kind, n being the number of earlier skeletons sharing the component, and $d(\mathbf{x})$ is a near-duplicate penalty. $\mathcal{C}(\mathbf{x}) = \min(1, 0.1 \max(0, |\mathbf{x}| - 4))$ charges atoms beyond the minimal four. The frontier term

$$\mathcal{F}(\mathbf{x}) = 4\mathcal{J}_{\mathbf{x}}(1 - \mathcal{J}_{\mathbf{x}}) \cdot \frac{\max(0, \mathcal{J}_{\mathbf{x}} - \mathcal{J}_{\mathbf{x}}^0)}{\mathcal{J}_{\mathbf{x}}} \quad (8)$$

is positive exactly on $\mathcal{B}_{\mathbf{e}}$ with $\mathcal{F} = 0$ when $\mathcal{J}_{\mathbf{x}} = 0$. The first factor is the normalized variance of the final success, maximal where the policy succeeds half of the time; the second is the share of that success acquired by training. This frontier reward follows the curriculum-learning principle of targeting tasks at intermediate difficulty [26, 28, 32, 33] and is maximized by tasks that training makes learnable but does not master. The positive terminal reward for the GFlowNet is

$$R(\mathbf{x}) = \epsilon + \exp(\beta \mathcal{V}(\mathbf{x})), \quad (9)$$

where $\epsilon > 0$ ensures positivity and β controls reward sharpness. For fairness, every baseline method in Sec. 5 is scored with the same $\mathcal{V}$.

4.3 FOSA Task Policy

Task State, Action, and Reward. For each compiled environment $\mathcal{M}_{\mathbf{e}, \mathbf{x}}$, FOSA trains a model-free task policy

$$\pi_{\mathbf{x}} = \arg \max_{\pi} \mathbb{E}_{\mathcal{M}_{\mathbf{e}, \mathbf{x}}, \pi} \left[\sum_{h=0}^H \gamma^h r_{\mathbf{e}, \mathbf{x}}(\mathbf{o}_h, \mathbf{a}_h) \right]. \quad (10)$$

The observation and action spaces follow the robot. The LLM compiler designs the state observed by the actor, the privileged state observed by the critic, the reward $r_{\mathbf{e}, \mathbf{x}}$, and the domain randomization, which exposes the policy to a rich range of dynamics for robust sim-to-real transfer.

Language-Conditioned Policy. The discovery stage produces task policies paired with natural language descriptions. FOSA distills them into a single language-conditioned student policy

$\pi_\phi(a \mid \mathbf{o}, \mathbf{x}_{\text{Lang}})$: a transformer over the observations and the task text, where $\mathbf{x}_{\text{Lang}}$ is encoded by the pretrained language backbone of MiniVLA [53]. The student is trained with data aggregation (Dagger) [54] on every trained task of a method’s archive: it is rolled out in the archived environments, and the corresponding teacher $\pi_\mathbf{x}$ labels visited observations with corrective actions. This produces one policy interface while preserving the embodiment-grounded supervision obtained during discovery. The language backbone is frozen and only the transformer and action head are trained.

Running Example. For a quadruped with $\mathbf{x}^0 = \emptyset$, the abstract policy may add a robot entity to $E_\mathbf{x}$, add an `inclined_ramp` asset, compose it with a `low_friction` constraint in $\Phi_\mathbf{x}$, refine $\theta_\mathbf{x}$ to a 20° incline, and terminate with a `track_velocity` goal in $G_\mathbf{x}$. The task is compiled into a slippery-slope environment, a PPO policy is trained, and the resulting $\mathcal{J}_\mathbf{x}$ updates the abstract policy.

5 Experimental Results

We evaluate FOSA on four heterogeneous robots in simulation and the real world: a ClearPath Jackal vehicle, an Inspire Dexterous hand, a Unitree Go2 quadruped, and a Booster T1 humanoid.

Under a fixed discovery budget, each method trains 256 tasks per robot in one discovery run, and the resulting task policies are distilled into a single language-conditioned policy. All methods use Gemma-4-31B-it for task verbalization and environment compilation, mujoco-playground [55] for simulation, and PPO [51] for task policies. The abstract reward uses $\beta = 8$, $\lambda_{\text{nov}} = 0.4$, $\lambda_{\text{cost}} = 0.05$, novelty weights $(w_{\text{atom}}, w_{\text{pair}}, w_{\text{sig}}) = (0.4, 0.4, 0.2)$, and a near-duplicate threshold of 0.85 atom-set Jaccard similarity; further details are in the supplementary material.

Research Questions and Metrics. [Q1] Does FOSA discover a lexically diverse task distribution? Div-SB4 and Div-SB5 are $1 - \text{Self-BLEU}$ over generated task descriptions [37, 56]; the 4-gram variant penalizes repeated phrases and the 5-gram variant repeated task templates, so higher is more diverse. [Q2] Does FOSA cover a broad semantic space? Dist-Embed1 and Dist-Embed2 measure word- and phrase-level dispersion in the language-embedding space [37, 56], and Expl-Eff is the fraction of candidates that yield executable policy-training runs. [Q3] Does FOSA generate tasks near each embodiment’s empirical capability frontier? Frontier-Prox is the mean $\mathcal{F}(\mathbf{x})$ (Eq. (8)) over generated tasks. [Q4] Does the discovered distribution improve downstream generalization? We measure the success and normalized episodic reward (shaped return divided by a task-specific scale) of the distilled policy on unseen tasks.

Held-Out Tasks. For each robot we author four held-out tasks with the LLM, disjoint from every method’s discovery log (Fig. 3): *uphill climb*, *ice patch crossing*, *gravel reverse*, and *tight turn* (vehicle); *large sphere lift*, *cuboid turn* (counterclockwise), *palm ball balance*, and *cylinder tip* (hand); *stair descent*, *weak front legs*, *push stand*, and *gentle ramp up* (quadruped); *single shove recovery*, and *crouch hold* (humanoid).

Baselines and Ablations. We compare against six task-discovery mechanisms. **LLM-CL** is an LLM curriculum that proposes each next task from the task history [37, 40, 41]. **LLM-MCTS** branches a Monte Carlo tree search (MCTS) with LLM-generated symbolic expansions [57]; **MCTS** removes the LLM and expands directly from the symbolic abstraction library; **MCTS-Learn** adds MuZero-style learned representation, dynamics, reward, policy, and value models [58]. **FOSA-RND**

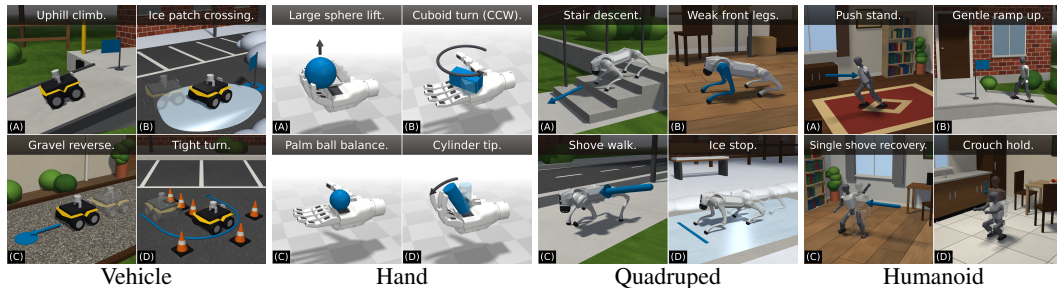


Figure 3: Held-out evaluation tasks for the distilled language-conditioned policy across four robots.

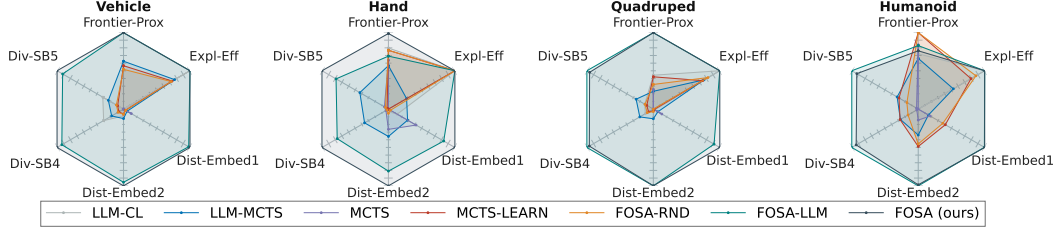


Figure 4: Comparison of discovered tasks along different axes: lexical diversity (Div-SB4, Div-SB5), semantic dispersion in the language-embedding space (Dist-Embed1, Dist-Embed2), exploration efficiency (Expl-Eff), and proximity to the empirical capability frontier (Frontier-Prox).

Table 1: Normalized reward of the distilled policy on held-out tasks.

Method	Vehicle				Hand				Quadruped				Humanoid				
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D	
Baseline	LLM-CL	0.64	0.36	0.57	0.76	0.29	0.91	0.00	0.57	0.33	0.77	0.65	0.28	0.88	0.12	0.11	0.11
	LLM-MCTS	0.48	0.36	0.74	1.00	0.61	0.00	0.00	0.75	0.51	0.18	0.97	0.18	0.18	0.13	0.71	0.06
	MCTS	0.62	0.35	0.86	0.85	0.14	1.00	0.27	0.61	0.28	0.42	0.97	0.29	0.71	0.95	0.93	0.71
	MCTS-Learn	0.58	1.00	0.31	1.00	0.49	0.15	0.00	0.57	0.30	0.52	1.00	0.22	0.88	0.97	0.79	0.98
Ablation	FOSA-RND	0.42	0.98	0.80	0.37	0.59	0.55	0.00	0.29	0.28	0.28	0.99	0.16	1.00	0.13	1.00	1.00
	FOSA-LLM	0.70	0.84	1.00	0.83	1.00	0.58	0.94	0.41	0.83	1.00	0.71	0.95	0.11	0.11	0.85	0.94
	FOSA (Ours)	1.00	0.69	0.57	0.87	0.49	0.15	1.00	1.00	1.00	0.97	0.95	1.00	0.97	1.00	0.87	0.82

A–D follow the held-out task order listed in Sec. 5 for each robot. Blue bold indicates the best method and black bold indicates the second best.

replaces the GFlowNet objective with random network distillation over symbolic abstractions [59], and **FOSA-LLM** replaces the learned abstract policy with LLM-proposed edits. Together they isolate the effects of symbolic search, LLM priors, and the learned abstract policy.

5.1 Simulation Results

Can FOSA discover diverse tasks efficiently? Fig. 4 compares discovered tasks across diversity, semantic coverage, exploration efficiency, and frontier proximity. Under the same 256-task budget, FOSA has the strongest overall profile on all robots. Its gains over the MCTS variants indicate that a learned stochastic construction policy is more effective than expanding a tree with hand-specified or value-estimated branching, and its gains over the LLM-driven baselines indicate that language-level plausibility is insufficient: *generated tasks must also be grounded in whether the embodiment can learn them under a fixed policy-training budget*. All LLM-driven baselines use one fixed model; a stronger language prior would complement, not replace, embodiment-grounded scoring.

Can FOSA approach empirical capability frontiers? FOSA discovers tasks closer to the empirical frontier on every robot except humanoid, where FOSA-RND scores marginally higher on Frontier-Prox. The gap reflects the larger and sparser symbolic space of whole-body humanoid control, in which an RND-style novelty bonus can occasionally reach frontier skeletons before the learned generator has accumulated reward feedback.

Tasks that appear semantically easy can be physically hard, and vice versa. For hand, a simple cube object is never reoriented even under loose orientation tolerance, whereas a ball of similar size is reoriented in more than half of the tasks. For vehicle, halving the wheel torque or adding more than 10 cm of terrain relief barely changes velocity tracking success, and for quadruped a 30 % torque loss matters only above 0.5 m s^{-1} . Such inversions are invisible to semantic plausibility and should be identified by grounding task generation in physical policy learning.

Can FOSA generalize to unseen tasks? Fig. 5 reports the distilled policy on the discovered tasks and Table 1 on the 16 held-out tasks. Averaged over held-out tasks, the policy distilled from FOSA-discovered tasks attains the highest normalized reward (0.83), ahead of FOSA-LLM (0.74) and the strongest baseline, MCTS (0.63), with the largest margins on quadruped and humanoid, where

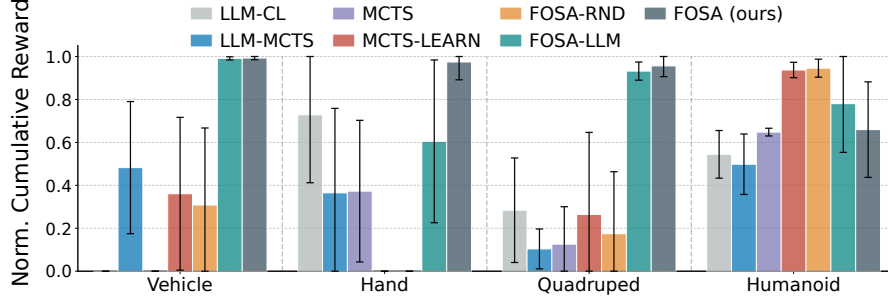


Figure 5: Simulation evaluation shows normalized cumulative reward of the distilled language-conditioned policies on the explored task suites. Each task is evaluated over 10 random seeds.

whole-body contact makes difficulty hardest to judge from language. The ablations sharpen this: FOSA-LLM, which keeps FOSA’s grammar and reward but replaces the learned P_F with LLM-proposed edits, matches or exceeds FOSA on `vehicle` and `hand` yet drops to 0.50 on `humanoid`. A language prior thus suffices where semantic and physical difficulty coincide and fails where they diverge, which is where the reward-grounded policy matters most. Consistent with Fig. 4, methods that discover diverse tasks near the frontier also provide more useful downstream supervision.

5.2 Algorithmic Analysis

We analyze FOSA’s sensitivity to its reward weights and discovery budget, and the convergence of its abstract policy (Fig. 6). Sensitivities are reported relative to FOSA’s default setting.

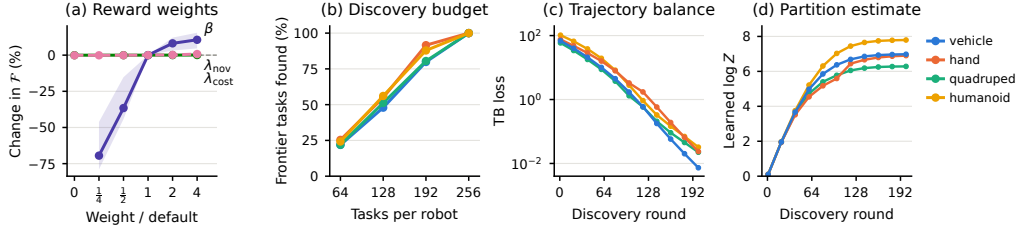


Figure 6: (a) Sensitivity to reward weights, as the change in expected frontier score of the target $P^* \propto R$. (b) Sensitivity to discovery budget, as the fraction of FOSA’s final frontier tasks found after its first n trained tasks. (c)–(d) GFlowNet training diagnostics over discovery rounds: trajectory-balance loss (log scale) and learned $\log Z$, mean over three seeds per robot.

Reward sensitivity. On the distinct skeletons trained by all methods except LLM-CL, we vary one weight of $\mathcal{V}$ at a time between $1/4\times$ and $4\times$ its default and track the expected frontier score of $P^*(\mathbf{x}) \propto R(\mathbf{x})$. Since $\mathcal{F}$ dominates $\mathcal{V}$, $\lambda_{\text{nov}} \in [0, 1.6]$ and $\lambda_{\text{cost}} \in [0, 0.2]$ move the score by at most 2% and 0.2%. Only β matters: halving it lowers the score by 15% to 44% and quartering it by 46% to 79%, leaving 29% to 65% of the target’s mass on the frontier instead of 98% to 100%; quadrupling it gains at most 16% but drops over a third of the goal families on quadruped. The default is thus the smallest β that concentrates on the frontier without losing coverage.

Budget sensitivity. Frontier tasks keep accumulating throughout discovery: after 64, 128, and 192 trained tasks, FOSA has found 22% to 25%, 48% to 56%, and 80% to 92% of its final frontier tasks across robots. Frontier membership is also decided early in each task’s training: rescore $\mathcal{F}$ after 56% of the PPO budget retains a rank correlation of 0.79 to 0.92 with the full-budget ranking and keeps 85% to 96% of frontier labels, so the per-task budget could be cut by roughly 40% without changing what FOSA discovers.

Convergence. On every robot, the trajectory-balance loss falls by four orders of magnitude over 200 discovery rounds and the learned $\log Z$ flattens after about 128. On the pooled replayed skeletons, the learned flow tracks the target reward (Spearman $\rho = 0.90$, Pearson $r = 0.98$, MAE 0.094 in $\log R$). These diagnostics verify the online objective empirically; the archive-dependent novelty term precludes a stationary-reward guarantee.



Figure 7: Held-out evaluation tasks for the distilled language-conditioned policy across two robots.

Method		Vehicle				Quadruped			
		A	B	C	D	A	B	C	D
Baseline	LLM-CL	0	50	0	17	0	0	0	33
	LLM-MCTS	17	33	0	17	0	0	0	0
	MCTS	0	33	17	17	0	0	0	0
	MCTS-Learn	17	50	17	33	0	0	0	0
Ablation	FOSA-RND	17	50	0	0	0	0	0	0
	FOSA-LLM	0	17	0	17	0	0	17	83
	FOSA (Ours)	50	67	17	33	17	0	17	100

Table 2: Real-world success rate (%) over six trials per held-out task.

5.3 Real-World Results

We test whether capabilities discovered in simulation transfer zero-shot to novel tasks on a ClearPath Jackal and a Unitree Go2, each running on an NVIDIA Jetson Orin with an Intel RealSense D435i at 30 Hz for policy inference and an OS1-64 LiDAR for 100 Hz odometry [60, 61]. Each robot has four held-out tasks (Fig. 7): *grass traversal*, *step climbing*, *obstacle bypass*, and *cart pushing* for vehicle; *grass traversal*, *barrier balance*, *mug delivery*, and *tunnel crawl* for quadruped.

Do discovered capabilities transfer to hardware? Table 2 reports success over six trials per method–task pair. The trend matches simulation: policies distilled from FOSA-discovered tasks transfer more reliably than those from baseline task sets. Absolute success is low for every method, which we attribute to the lightweight student and the restricted simulation training protocol (Sec. 6). The hardest task is *barrier balance*: narrow support, low-friction contact, and posture transitions amplify errors the simulator does not fully reproduce. Within these constraints, FOSA achieves the highest real-world success, supporting capability-frontier discovery as a practical source of transferable robot training.

These results support FOSA’s central premise. Because the task space is combinatorial, discovery must allocate its budget to tasks that are both learnable and informative, and the two failure modes are asymmetric: a task already solved at initialization adds no behavioral coverage, whereas a task still unsolved after training yields a teacher whose actions are uninformative or inconsistent. By sampling in proportion to a reward that is positive only on the frontier, FOSA fills its archive with tasks whose teachers encode nontrivial yet reliable control, the regime in which distillation is most sample-efficient. The archive thereby contains transferable subskills (e.g., balance regulation, multi-stage navigation) that the student recombines on tasks never seen during discovery, so the gains on unseen tasks reflect better transfer of discovered capabilities as well as better discovery.

6 Conclusion and Limitations

We presented Generative Flow on Symbolic Abstractions (FOSA), a framework for discovering embodiment-specific capabilities by learning an abstract policy over symbolic task skeletons. FOSA compiles generated skeletons into simulation environments and uses low-level policy-learning performance as feedback to discover tasks near the robot’s empirical capability frontier. Across four heterogeneous robots, FOSA improves discovery efficiency, task diversity, and frontier proximity, and the policy distilled from its tasks executes unseen instructions on real hardware.

Limitations. *Empirical frontier.* The frontier depends on the simulator, policy class, reward design, and training budget, although its ranking is stable across budgets (Sec. 5.2). *Sim-to-real.* The student is a lightweight language-conditioned transformer rather than a full VLA, and compute limits kept domain randomization and task training below those of dedicated sim-to-real pipelines [4, 5]. *Safety.* FOSA targets tasks at the edge of competence, so rollouts risk falls and collisions; all hardware runs used an operator-held emergency stop and joint velocity limits, and automatic safety filtering of generated tasks is left for future work.

References

- [1] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.
- [2] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.
- [3] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*, 2018.
- [4] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science robotics*, 4(26):eaau5872, 2019.
- [5] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. *arXiv preprint arXiv:2107.04034*, 2021.
- [6] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.
- [7] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.
- [8] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.
- [9] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4(1):265–293, 2021.
- [10] B. Li, T. Silver, S. Scherer, and A. Gray. Bilevel Learning for Bilevel Planning. In *Proceedings of the Robotics: Science and Systems (RSS)*, 2025.
- [11] Y. Huang, B. Li, V. Saxena, Y. Liang, U. Mishra, L. Ji, L. Zha, J. Wu, N. Kumar, S. Scherer, D. Xu, and T. Silver. Kinder: A physical reasoning benchmark for robot learning and planning. In *Robotics: Science and Systems (RSS)*, 2026.
- [12] Y. Bengio, S. Lahlou, T. Deleu, E. J. Hu, M. Tiwari, and E. Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.
- [13] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.
- [14] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.
- [15] K. Pertsch, Y. Lee, and J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021.
- [16] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.
- [17] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu. Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization. *arXiv preprint arXiv:2111.07999*, 2021.

- [18] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik’s cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.
- [19] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.
- [20] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.
- [21] C. Bao, H. Xu, Y. Qin, and X. Wang. Dexart: Benchmarking generalizable dexterous manipulation with articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21190–21200, 2023.
- [22] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.
- [23] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giró-i Nieto, and J. Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International conference on machine learning*, pages 1317–1327. PMLR, 2020.
- [24] M. Laskin, H. Liu, X. B. Peng, D. Yarats, A. Rajeswaran, and P. Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.
- [25] H. Liu and P. Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.
- [26] C. Florensa, D. Held, X. Geng, and P. Abbeel. Automatic goal generation for reinforcement learning agents. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1515–1528. PMLR, 10–15 Jul 2018.
- [27] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep rl: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.
- [28] Y. Zhang, P. Abbeel, and L. Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.
- [29] R. Wang, J. Lehman, J. Clune, and K. O. Stanley. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753*, 2019.
- [30] M. Dennis, N. Jaques, E. Vinitzky, A. Bayen, S. Russell, A. Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *Advances in neural information processing systems*, 33:13049–13061, 2020.
- [31] M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4940–4950. PMLR, 18–24 Jul 2021.
- [32] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning*, pages 17473–17498. PMLR, 2022.
- [33] S. Tio and P. Varakantham. Transferable curricula through difficulty conditioned generators. *arXiv preprint arXiv:2306.13028*, 2023.

- [34] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Thompson, I. Mordatch, Y. Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*, 2022.
- [35] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng. Code as policies: Language model programs for embodied control. In *2023 IEEE International conference on robotics and automation (ICRA)*, pages 9493–9500. IEEE, 2023.
- [36] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang. Gensim: Generating robotic simulation tasks via large language models. In *Arxiv*, 2023.
- [37] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.
- [38] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, J. Fan, et al. Eureka: Human-level reward design via coding large language models. In *International conference on learning Representations*, volume 2024, pages 26516–26560, 2024.
- [39] T. Xie, S. Zhao, C. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu. Text2reward: Reward shaping with language models for reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pages 35663–35699, 2024.
- [40] Y. J. Ma, W. Liang, H. Wang, S. Wang, Y. Zhu, L. Fan, O. Bastani, and D. Jayaraman. Dreureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024.
- [41] W. Liang, S. Wang, H.-J. Wang, O. Bastani, D. Jayaraman, and Y. J. Ma. Environment curriculum generation via large language models. In *Conference on Robot Learning (CoRL)*, 2024.
- [42] H. Xia, X. Li, Z. Li, Q. Ma, J. Xu, M.-Y. Liu, Y. Cui, T.-Y. Lin, W.-C. Ma, S. Wang, et al. Sage: Scalable agentic 3d scene generation for embodied ai. *arXiv preprint arXiv:2602.10116*, 2026.
- [43] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.
- [44] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.
- [45] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. In P. Agrawal, O. Kroemer, and W. Burgard, editors, *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 2679–2713. PMLR, 06–09 Nov 2025.
- [46] M. J. Kim, C. Finn, and P. Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.
- [47] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik. Robogrammar: graph grammar for terrain-optimized robot design. *ACM Trans. Graph.*, 39(6), Nov. 2020. ISSN 0730-0301.
- [48] Y. Yuan, Y. Song, Z. Luo, W. Sun, and K. Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. *arXiv preprint arXiv:2110.03659*, 2021.
- [49] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei. Metamorph: Learning universal controllers with transformers. *arXiv preprint arXiv:2203.11931*, 2022.

- [50] N. Malkin, M. Jain, E. Bengio, C. Sun, and Y. Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35:5955–5967, 2022.
- [51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [52] M. Kim, T. Yun, E. Bengio, D. Zhang, Y. Bengio, S. Ahn, and J. Park. Local search GFlownets. In *The Twelfth International Conference on Learning Representations*, 2024.
- [53] S. Belkhale and D. Sadigh. Minivla: A better vla with a smaller footprint, 2024.
- [54] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.
- [55] K. Zakka, B. Tabanpour, Q. Liao, M. Haiderbhai, S. Holt, J. Y. Luo, A. Allshire, E. Frey, K. Sreenath, L. A. Kahrs, C. Sferrazza, Y. Tassa, and P. Abbeel. Demonstrating MuJoCo Playground. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025.
- [56] Y. Zhu, S. Lu, L. Zheng, J. Guo, W. Zhang, J. Wang, and Y. Yu. Taxygen: A benchmarking platform for text generation models. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 1097–1100, 2018.
- [57] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.
- [58] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.
- [59] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.
- [60] C. Bai, T. Xiao, Y. Chen, H. Wang, F. Zhang, and X. Gao. Faster-LIO: Lightweight Tightly Coupled Lidar-Inertial Odometry Using Parallel Sparse Incremental Voxels. *IEEE Robotics and Automation Letters*, 7(2):4861–4868, 2022.
- [61] Y. Yu, Y. Liu, F. Fu, S. He, D. Zhu, L. Wang, X. Zhang, and J. Li. Fast extrinsic calibration for multiple inertial measurement units in visual-inertial system. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 01–07, 2023.

Generative Flow on Symbolic Abstractions for Embodiment Capability Discovery

Supplementary Material

Youwei Yu Lantao Liu

Indiana University Bloomington
 {youwyu, lantao}@iu.edu

A1 Symbolic Task Skeleton Language and Notation

This section expands the symbolic task representation used by FOSA. We use the same notation as the main paper: a task skeleton is $\mathbf{x} = [E_{\mathbf{x}}, I_{\mathbf{x}}, G_{\mathbf{x}}, U_{\mathbf{x}}, \theta_{\mathbf{x}}, \Phi_{\mathbf{x}}]$, where $E_{\mathbf{x}}$ is the set of typed entities, $I_{\mathbf{x}}$ is the initial symbolic state, $G_{\mathbf{x}}$ is the goal condition, $U_{\mathbf{x}}$ is the set of referenced operator schemas, $\theta_{\mathbf{x}}$ stores task arguments, and $\Phi_{\mathbf{x}}$ stores feasibility constraints. A partial skeleton $\mathbf{x}^t$ has the same structure, but some fields may be empty or only partially specified.

Skeleton Components. The symbolic domain is specified by entity types $\mathcal{Y}$, predicate schemas $\mathcal{P}$, operator schemas $\mathcal{U}$, and feasibility constraints Φ . Given a finite entity set E , the predicate schemas induce ground atoms $\text{Atoms}(\mathcal{P}, E)$ and literals $\text{Lits}(\mathcal{P}, E)$. The fields of a skeleton are summarized in Table A1.1.

Table A1.1: Components of a symbolic task skeleton.

Field	Meaning	Examples
$E_{\mathbf{x}}$	Typed entities in the task.	robot, target, object, payload, ramp, obstacle, terrain patch, door, tool
$I_{\mathbf{x}}$	Initial symbolic state induced by the reset distribution.	initial robot pose, object pose, terrain condition, contact relation, visibility condition
$G_{\mathbf{x}}$	Goal literals or success conditions.	reach target, track velocity, lift object, maintain balance, rotate object, avoid obstacle
$U_{\mathbf{x}}$	Operator or task-family schemas referenced by the skeleton.	navigate, climb, push, grasp, lift, rotate, balance, transport
$\theta_{\mathbf{x}}$	Continuous or discrete arguments needed to instantiate the task.	slope angle, friction, object mass, target distance, velocity command, tolerance, duration
$\Phi_{\mathbf{x}}$	Feasibility, safety, and simulator constraints.	valid pose ranges, contact constraints, termination thresholds, admissible friction ranges, collision constraints

A skeleton is not only a semantic description; it must also be supported by the embodiment and back-end. For example, a quadruped can instantiate terrain, balance, and locomotion-related skeletons, while a dexterous hand can instantiate grasping, lifting, and in-hand manipulation skeletons.

Partial Skeletons and Construction. FOSA constructs skeletons through a sequence of symbolic edits. The initial state is the empty skeleton $\mathbf{x}^0 = \emptyset$, and a construction trajectory is $\tau = (\mathbf{x}^0 \rightarrow \mathbf{x}^1 \rightarrow \dots \rightarrow \mathbf{x}^T = \mathbf{x})$. Each intermediate state $\mathbf{x}^t$ is a partial skeleton. A partial skeleton may already contain some entities, predicates, goals, operators, arguments, or constraints, but it is not evaluated until it becomes complete.

A complete skeleton must contain enough information to define an RL environment: it must specify the relevant entities, a nonempty goal condition, any operator schemas needed to interpret the goal, all required task arguments, and the constraints needed to instantiate the reset distribution, reward, success predicate, and termination conditions. The argument vector $\theta_{\mathbf{x}}$ contains task-specific parameters. Different task families expose different arguments. For example, locomotion tasks may use slope, roughness, and commanded velocity, while manipulation tasks may use object size, object mass, friction, and rotation angle.

Validity and Completeness. The validity predicate checks whether a partial skeleton is syntactically well-formed and supported by the robot and simulator:

$$\text{Valid}_{\mathbf{e}, \mathcal{M}}(\mathbf{x}^t) = \mathbb{I} \left[\begin{array}{l} E_{\mathbf{x}^t} \text{ uses supported entity types,} \\ I_{\mathbf{x}^t} \subseteq \text{Atoms}(\mathcal{P}, E_{\mathbf{x}^t}), \\ G_{\mathbf{x}^t} \subseteq \text{Lits}(\mathcal{P}, E_{\mathbf{x}^t}), \\ U_{\mathbf{x}^t} \text{ uses supported operator schemas,} \\ \theta_{\mathbf{x}^t} \text{ lies in admissible domains,} \\ \Phi_{\mathbf{x}^t} \text{ uses supported feasibility constraints.} \end{array} \right].$$

The completeness predicate checks whether a valid partial skeleton contains all information needed for environment compilation:

$$\text{Complete}_{\mathbf{e}, \mathcal{M}}(\mathbf{x}^t) = \mathbb{I} \left[\begin{array}{l} \text{Valid}_{\mathbf{e}, \mathcal{M}}(\mathbf{x}^t) = 1, \\ G_{\mathbf{x}^t} \neq \emptyset, \\ \text{all required arguments in } \theta_{\mathbf{x}^t} \text{ are specified,} \\ \text{a reset distribution, reward, success predicate, and termination} \\ \text{condition can be induced from } \mathbf{x}^t. \end{array} \right].$$

Canonical Form and Duplicate Structure. Different construction orders can lead to the same skeleton. We identify a skeleton by the sorted multiset of its bindings (atom, role, and arguments), which removes construction-order artifacts. All reported metrics count a skeleton once under this identity. FOSA’s sampler tracks skeletons by construction path, so local search can re-derive and re-train an earlier skeleton (Sec. 4.2). Before compilation, every grammar-based method rejects a skeleton whose key or description exactly matches an archived one. Let $S(\mathbf{x})$ be the set of atoms of $\mathbf{x}$. Near-duplicates, with

$$\text{Jac}(\mathbf{x}, \mathbf{y}) = \frac{|S(\mathbf{x}) \cap S(\mathbf{y})|}{|S(\mathbf{x}) \cup S(\mathbf{y})|} \geq 0.85 \quad (\text{A1.1})$$

against a recent skeleton $\mathbf{y}$ and no changed numeric argument, are not rejected but reduce the novelty term $\mathcal{N}$, so meaningful variations in difficulty remain available.

A2 Experiment Results

This section first provides details for the baseline approaches and the 16 held-out evaluation tasks. It then shows additional results from simulation and hardware transfer.

A2.1 Baseline Implementation Details

LLM-CL. LLM-CL is a direct language-model curriculum baseline. At each iteration, the LLM receives the robot and backend capability context, together with previous task descriptions and policy performance history. It then proposes one concrete next task.

LLM-CL prompt template.**System message:**

You design feasible robot reinforcement-learning curricula for the active simulator backend. Output JSON only.

User message: a JSON object with the following fields:

- `backend`, `robot`, `robot_display_name`, `generator_robot_name`: identifiers for the active robot.
- `robot_profile`, `backend_capabilities`, `simulator_capability_manifest`: embodiment description and the simulator contract.
- `registered_task_adapters`, `backend_supported_families`, `applicable_task_families`, `unsupported_families_with_reasons`: what the backend can instantiate.
- `objective`: “Propose the next task at the robot’s capability frontier: one it cannot do yet (success_rate near 0 at the start of training) but can learn within the budget (success_rate between 0.1 and 0.9 at the end). Tasks already solved, and tasks that stay at zero, are both worthless.”
- `history`: the last 20 tasks, with description, status, success rate at the start and end of training, normalized reward, $\mathcal{F}$, and errors, including rejected proposals.
- `previous_validation_issue`: the reason the previous proposal was rejected, if any.

If the prompt is too long, the profile and manifest, then the capability lists, are dropped, and the history is shortened to 10 and then 5 tasks.

Instructions: propose exactly one next concrete robot task; use only backend-supported capabilities and the simulator manifest; select `task_id` from the registered adapters; describe only the goal and the objects physically needed; vary physical context only when it changes the learning objective or is supported by recent outcomes; return JSON only.

Expected output schema: `task_id`, `execution`, `description`, `module_name`, `rationale`, `expected_difficulty`, `family_hint`.

MCTS. MCTS searches the same atom grammar as FOSA without LLM-based branching. Tree nodes are partial skeletons; the root’s children add one capability atom per execution primitive.

- **Selection.** UCT with score $\bar{v} + 1.4\sqrt{\ln(N+1)/(n+1)} + 0.25\mathbb{I}[n=0] + 0.15 \cdot 4\bar{p}(1-\bar{p})$, where $\bar{p}$ is the node’s mean success rate, and progressive widening admits a new child while the number of children is below $2\sqrt{N+1}$.
- **Expansion.** Up to four children: an atom added to an open slot, or a refinement that moves one numeric argument by one of five levels or sets a categorical argument to another admissible value.
- **Simulation.** A complete skeleton is verbalized, compiled, and trained with the same PPO budget as every method.
- **Backpropagation.** The node’s utility $\mathcal{V}$ (Eq. (7)), or -0.25 for a run that produced nothing, is added to every node on the path.

LLM-MCTS. The LLM proposes eight child skeletons for the selected node, given the parent, the grammar atoms, the most valuable atoms, and recent and failed tasks. Atom names are snapped to the nearest grammar atom, and the four highest-scoring valid children are kept. The LLM’s own description is used as the task text.

MCTS-Learn. MuZero-style representation, dynamics, reward, policy, and value networks (128-dimensional latent) are trained on the tree’s statistics with a one-step unroll. They act over eleven edit-operator types, and a 32-simulation latent search supplies a prior and value that are combined with the empirical node values.

FOSA-RND and FOSA-LLM. Both sample from $\mathbf{x}^0$ without training. FOSA-RND draws every edit uniformly among valid edits; FOSA-LLM adds FOSA’s LLM edit prior to the uniform logits.

A2.2 Real-World Results

The real-world evaluation covers the two available hardware platforms: the ClearPath Jackal vehicle and the Unitree Go2 quadruped. The hand and humanoid experiments are evaluated in simulation only. Although recent progress in sim-to-real transfer makes hardware deployment of these policies plausible, a detailed real-world evaluation of these embodiments is left for future work.

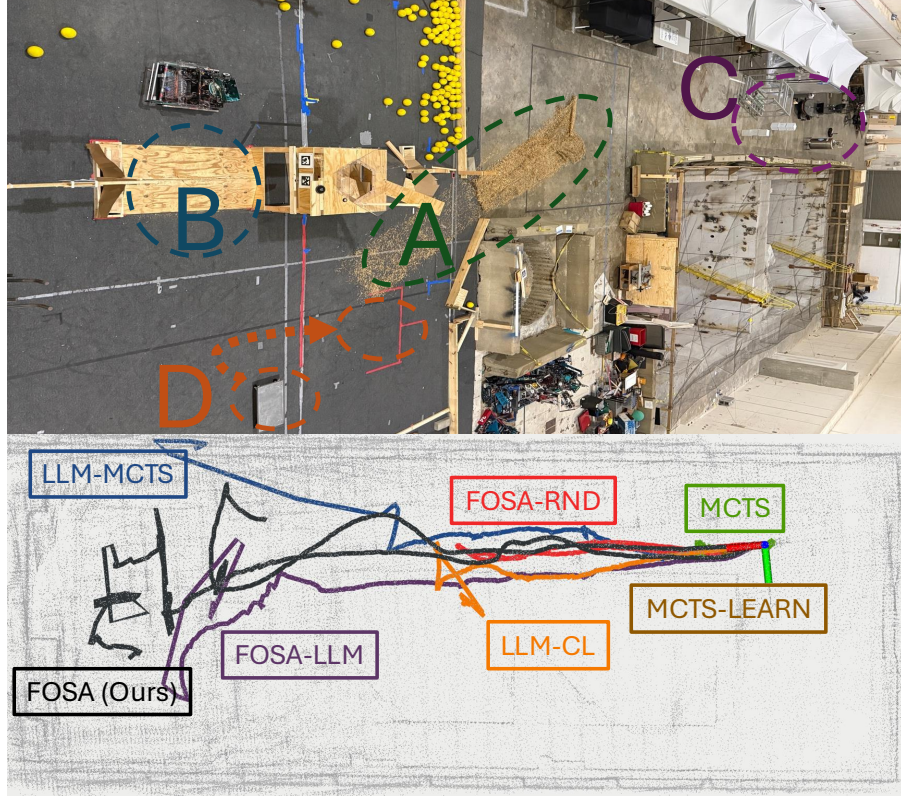


Figure A2.1: Real-world vehicle trajectories for FOSA, baselines, and ablations.

Vehicle. Fig. A2.1 shows the real-world vehicle trajectories of all compared methods in the reconstructed pointcloud map. The route consists of four stages: task C, obstacle bypass; task A, grass traversal; task B, step climbing; and task D, cart pushing. FOSA is the only method that completes the full route. The baseline methods usually fail before reaching task A, often due to obstacles or scene configurations that are not well covered during capability discovery. This trend is consistent with the main-paper results: baselines either discover less diverse task sets or fewer tasks near the empirical capability frontier.

The final cart-pushing stage is especially challenging for the ablations because the cart introduces nontrivial wheel dynamics and delayed contact effects. Although FOSA succeeds, the rollout also exposes limitations of the distilled language-conditioned policy: the vehicle sometimes produces jerky motions in the unstructured environment. Overall, these results suggest that FOSA benefits from discovering diverse, frontier-proximal capabilities, while mobile robot policies still face challenges when interacting with complex real-world objects.

Quadruped. In the grass-traversal task, most methods struggle with uneven terrain and traction variation, indicating that language-level task plausibility alone does not reliably capture the physical difficulty of the scene. In the barrier-balance task, no method fully completes the task, but FOSA makes the most progress: the policy is able to maintain balance on the narrow barrier without immediately falling off.

The mug-delivery task is intuitively simple, but it requires the robot to stabilize a carried object while traversing uneven terrain. Only FOSA and its ablation FOSA-LLM keep the mug stable through the traversal; the baselines lose it at the first terrain transition. In the tunnel crawl task, all methods can pass through the tunnel, but only FOSA and its ablation lower the body height as intended. We also observe that the FOSA policy can pass the tunnel using a reverse-motion strategy, suggesting that the distilled policy can exploit non-obvious behaviors discovered during capability generation.

Computation Time. Each task-policy run trains for 2×10^8 environment steps on 4096 parallel environments. With four runs sharing one GPU and four GPUs per discovery job, a run takes a median of 1.1 h (quadruped), 1.4 h (vehicle), 2.1 h (hand), and 2.2 h (humanoid) of wall-clock time. Distilling one student takes about 13 h to 19 h on four GPUs.

A2.3 Simulation Results

Policy Hyperparameters. We use three policy-learning components: the GFlowNet abstract policy for skeleton generation, PPO task policies for grounding each compiled environment, and a MiniVLA-based language-conditioned student policy for distillation.

Table A2.1: Hyperparameters of the three learned components.

Hyperparameter	Value
FOSA Abstract Policy (GFlowNet)	
Maximum construction depth $T_{\max}$	10
State encoder	Qwen2.5-0.5B
P_F, P_B architecture	MLP (256, 256)
Optimizer	Adam; lr 10^{-3} (P_F, P_B), 10^{-1} (log Z)
Replay buffer	1024
Replay batch size	32
Training updates per round	300
Reward sharpness β	8.0
Reward offset ϵ	10^{-4}
Utility weights	$\lambda_{\text{nov}} = 0.4, \lambda_{\text{cost}} = 0.05$
Near-duplicate threshold	atom Jaccard 0.85
FOSA Task Policy (PPO)	
Parallel environments	4096
Environment steps per task	2×10^8
Rollout length	20 steps (hand: 40)
Batch size; mini-batches	256; 32
Learning epochs per update	4
Learning rate	3×10^{-4}
Discount factor γ	0.97 (hand reorientation: 0.99)
GAE parameter λ	0.95
Clipping parameter	0.2
Value loss coefficient	0.5
Entropy coefficient	10^{-2} (vehicle, humanoid: 5×10^{-3})
Max gradient norm	1.0
Actor / critic	MLP (512, 256, 128)
Activation	swish
Episode length	1000 (hand reorientation: 600)
Language-conditioned Policy (MiniVLA)	
Policy	Transformer, 3 layers, 8 heads, $d = 256$
Inputs	last 10 actor observations + task text
Text encoder	Qwen2.5-0.5B
Action chunk	1
Dagger	1024 environments
Loss	Smooth- ℓ_1 imitation loss
Optimizer	AdamW, lr 3×10^{-4} , weight decay 10^{-4}
Batch size	4096 per GPU (micro-batch 512)

Training Logs. Figs. A2.2, A2.3, A2.4, and A2.5 show PPO training curves and simulation rollout snapshots for the four robot embodiments. These logs illustrate the task-policy learning process used to ground generated skeletons before archive construction and policy distillation.



Figure A2.2: Simulation vehicle learning snapshots with task descriptions and reward curves.

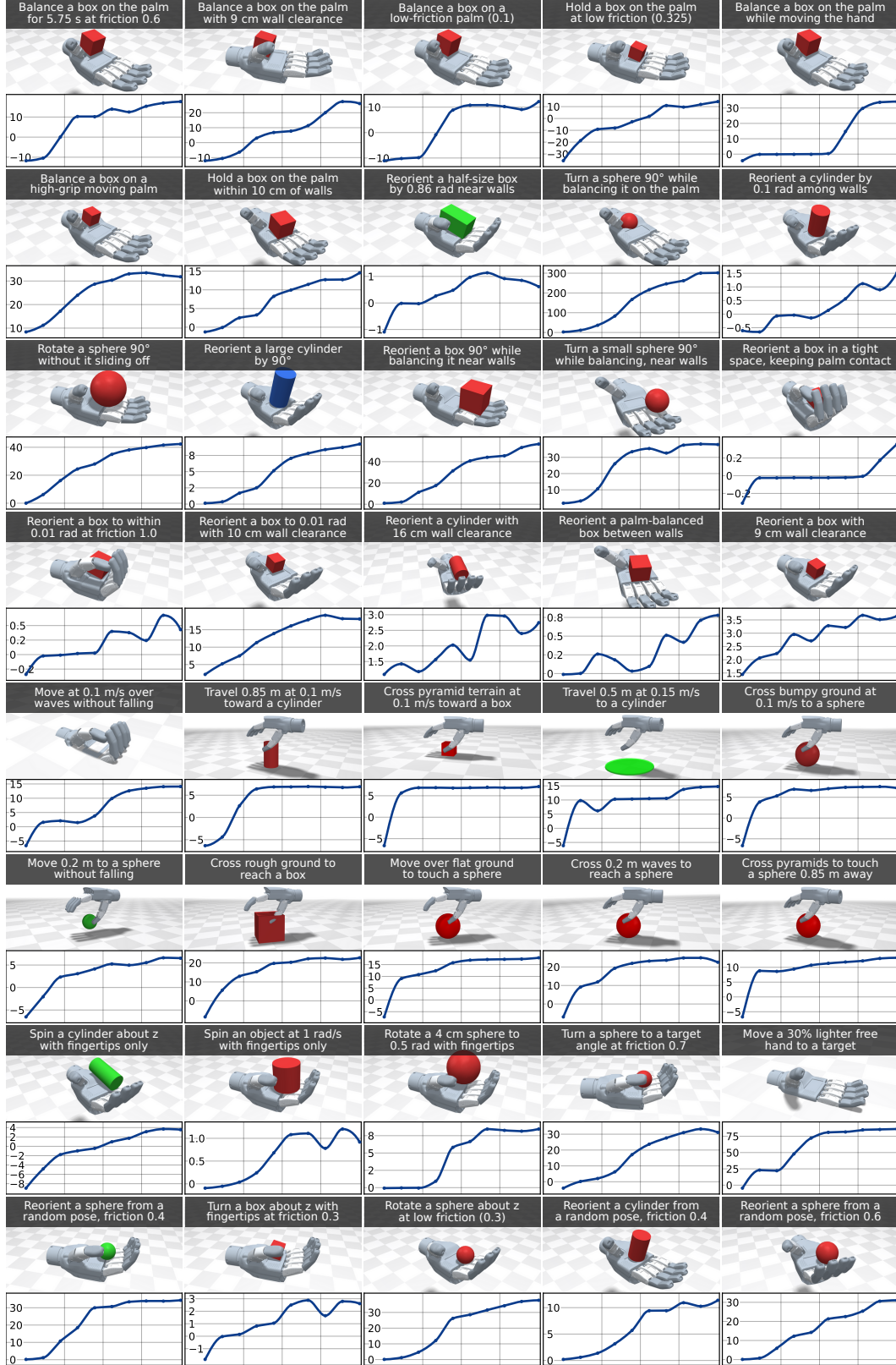


Figure A2.3: Simulation hand learning snapshots with task descriptions and reward curves.

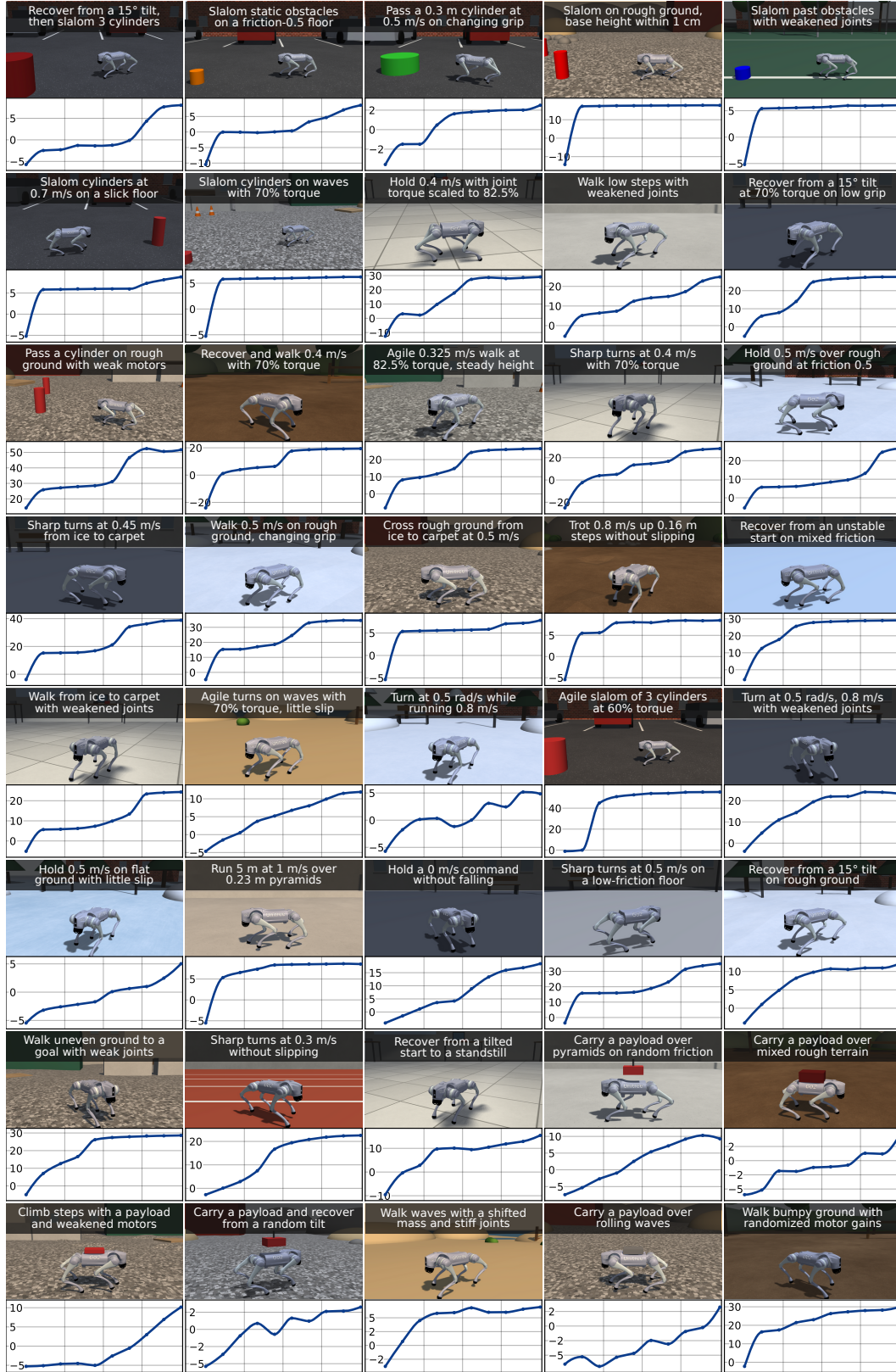


Figure A2.4: Simulation quadruped learning snapshots with task descriptions and reward curves.

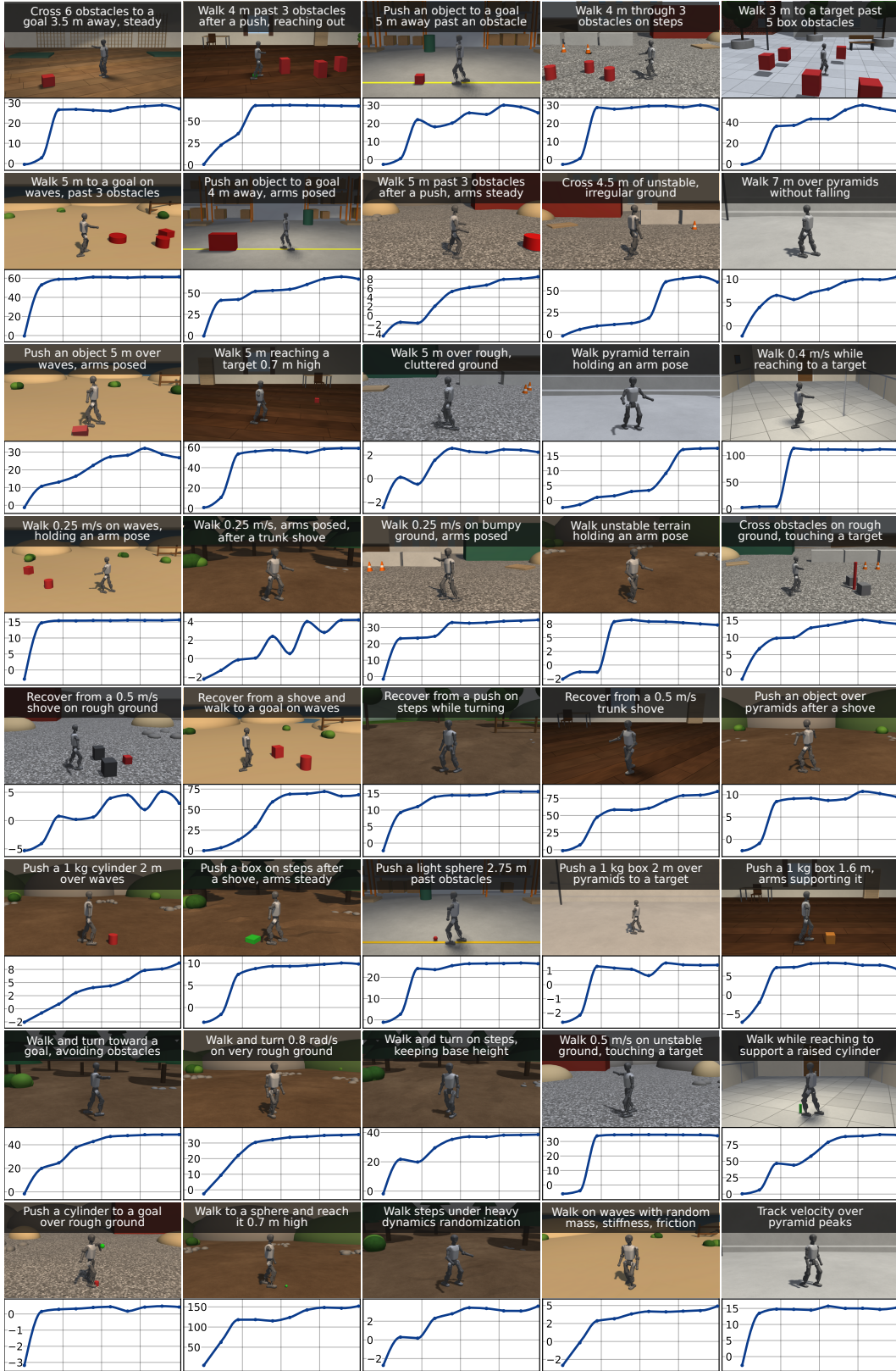


Figure A2.5: Simulation humanoid learning snapshots with task descriptions and reward curves.