SENSOR GENERALIST POLICIES FOR FAST OPTIMIZATION OF ROBOTIC SENSOR SUITES

Anonymous authors
Paper under double-blind review

ABSTRACT

While data-driven policy synthesis techniques have received much attention in robotics, sensor suites are still largely designed through human intuition and costly trial-and-error. This choice is especially consequential as robot learning scales: sensors determine what information is captured in the experience used for learning and what feedback is available for control. We formulate sensor selection as jointly searching for a policy and its associated sensor specification that maximize downstream task performance, across different sensing levels. Evaluating candidate configurations by independently training policies, however, is prohibitively expensive across a combinatorial search space. We address this bottleneck with the *Sensor Generalist*, a single policy trained in simulation across diverse sensor configurations that amortizes policy learning across sensing budgets. The generalist enables lightweight evaluation and search, yielding high-performing policies and their associated sensor specifications at different sensing levels. We analyze the performance of this approach for real-world and simulated dexterous manipulation with proprioception and tactile sensors, and quadruped locomotion with LiDAR-based obstacle avoidance in simulation. Our approach matches or outperforms the evaluated baselines while reducing training cost, and discovers non-intuitive, often compact sensor configurations that support effective learned behaviors, yielding interesting insights into the sensory requirements of these tasks. Website

1 INTRODUCTION

Any robot must sense the world in order to act in it. Cameras reveal objects and geometry, tactile sensors reveal forces and contact, and proprioception reveals the robot's own configuration. The choice of sensors therefore fundamentally determines what information is available for learning and control. This interface is especially consequential in the modern scaling paradigm of robot learning, where progress increasingly comes from scaling models, datasets, and robot experience. Yet while substantial effort has gone into learning how to act from increasingly large datasets, what the robot should sense is largely decided through human engineering. Simply instrumenting every possible sensor is also impractical due to associated power, computation, and costs. Therefore, we ask *which sensing capabilities are most valuable for the tasks a robot must perform?*

Sensor selection for control is challenging because the utility of a sensor depends on the behavior of the policy that uses it. Removing a sensor changes the observation process and can change the optimal behavior itself: a limited-view camera may suffice if the robot learns to reposition it, while a tactile sensor may become valuable only after the controller learns a strategy that exploits it. Prior work makes sensor selection tractable in several ways: by assuming structured or known system models that permit sensor utility to be computed directly (Joshi & Boyd, 2008); by scoring or removing sensors relative to a fixed learned model or controller (Kolter & Ng, 2009; Nguyen et al., 2013; Morimoto et al., 2008; Krishna et al., 2025; Wei et al., 2022); or by retraining policies for candidate sensor subsets (Kim & Ha, 2021). These approaches either impose strong assumptions, risk undervaluing sensors whose usefulness emerges only after behavior adapts, or incur the high cost of repeatedly solving new control problems.

We instead seek to evaluate sensors according to downstream control performance while retaining the flexibility of end-to-end policy learning and avoiding repeated policy optimization for every sen-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

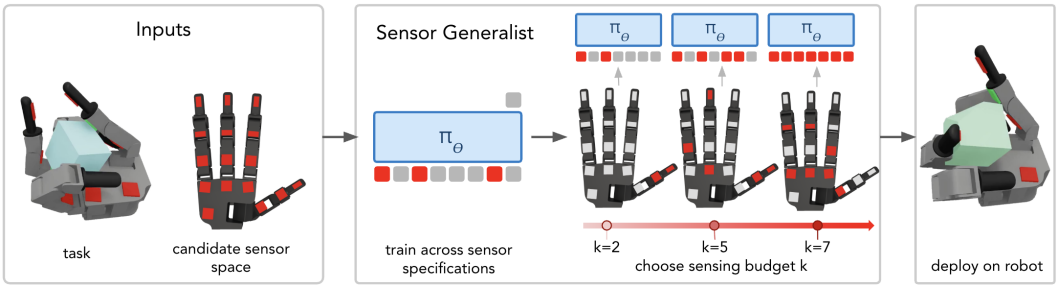


Figure 1: **Problem Setup.** A generalist policy is trained across varying sensor specifications, such as different placements of tactile sensors (red squares) on a dexterous hand. The generalist policy enables search over different sensor specifications across sensing levels, and the best ones are deployed on the robot.

sor suite. To do so, we introduce the Sensor Generalist (see fig. 1 for a visual overview), a single policy trained across a distribution of sensor specifications. Rather than learning an independent controller for each candidate sensor suite, the generalist shares learning across many partially observed control problems and can be queried under different sensor subsets without retraining. Conditioning the generalist on a sensor specification yields a controller that operates with those sensors. Evaluating its return therefore enables lightweight search for high-performing policy-sensor pairs across sensing budgets. We validate the Sensor Generalist on both real and simulated tasks, ranging from dexterous manipulation tasks with tactile sensing to quadruped navigation with range sensing. The results show the generalist enables efficient sensor selection while matching or outperforming large collections of separately trained specialist policies and alternative selection approaches.

Our contributions are: (1) formulation of robotic sensor selection as a joint search problem over policies and their associated sensor specifications over different sensing levels; (2) the Sensor Generalist, which amortizes policy learning across sensor configurations and enables inexpensive evaluation and search; and (3) an empirical study across diverse robotic control domains demonstrating efficient sensor selection and substantial reductions in training cost compared to alternative approaches.

2 Sensor Selection through Downstream Policy Performance

Our goal is to select sensors that enable a robot to perform its tasks effectively, while adhering to practical constraints like cost, weight, computation, energy, and more. In this work, we consider the number of sensors as the constraint, as it serves as a rough proxy for other practical constraints. The central question of sensor selection is: *how should we evaluate candidate sensor specifications?* Related feature-selection problems have been studied extensively in supervised learning using continuous relaxations, sparsity-inducing regularization, and feature-importance criteria (Joshi & Boyd, 2008; Louizos et al., 2018; Gu et al., 2011; Robnik-Šikonja & Kononenko, 2003; Chen & Guestrin, 2016; Breiman, 2001). In control and reinforcement learning (RL), prior work makes sensor selection tractable through structured estimation or control models or by scoring sensors relative to a fixed learned model or controller (Kolter & Ng, 2009; Nguyen et al., 2013; Morimoto et al., 2008; Krishna et al., 2025; Wei et al., 2022; Kroon & Whiteson, 2009; Parr et al., 2008; Hachiya & Sugiyama, 2010; Kishima & Kurashige, 2013; Brace & Morgansen, 2024). Such approaches depend heavily on strict assumptions and known models, and can rule out sensors whose usefulness only emerges after a policy adapts to them. Other works train policies for candidate sensor specifications using end-to-end deep RL (Kim & Ha, 2021; Guo et al., 2024). This allows behavior to adapt to available observations and evaluates sensors through the task performance of policies trained to use them, but the expense of training policies limits their search space and task complexity.

Despite the computational overhead, this perspective captures the criterion we seek: a sensor specification's value lies in the task performance it enables a policy to achieve. Because this performance depends on how the policy uses the available observations, we formulate sensor selection as a joint

search for **policies and their associated sensor specifications that maximize task return, over multiple levels of sensing**.

We model the process of sensor selection using the Partially Observed Markov Decision Process (POMDP). This is defined as $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, \Omega, R, \gamma)$. $\mathcal{S}$ is the hidden state space, $\mathcal{A}$ the action space, and $\mathcal{O}$ is the observation space; the dynamics are described by the transition density $T(s_{t+1} \mid s_t, a_t)$, observation model $\Omega(\cdot \mid s_t)$, and reward density $R(r_t \mid s_t, a_t)$.

A core concept is the *sensor specification*. Let $\mathcal{C} = \{c_1, \ldots, c_K\}$ be the set of all candidate sensors, with readings $o_t^{c_i} \sim \Omega^{c_i}(\cdot \mid s_t)$. A sensor specification $\sigma = \{c_{i_1}, \ldots, c_{i_m}\} \subseteq \mathcal{C}$ selects the sensors available to the policy, yielding observations

$$o_t^\sigma = (o_t^{c_{i_1}}, \ldots, o_t^{c_{i_m}}). \tag{1}$$

The corresponding observation model factorizes as $\Omega^\sigma(o_t^\sigma \mid s_t) = \prod_{c_i \in \sigma} \Omega^{c_i}(o_t^{c_i} \mid s_t)$. Each specification induces a POMDP $\mathcal{P}_\sigma = (\mathcal{S}, \mathcal{A}, \mathcal{O}^\sigma, T, \Omega^\sigma, R, \gamma)$, preserving the task dynamics and reward while changing the observations available to the policy. We denote the set of admissible sensor specifications by $\Sigma \subseteq 2^{\mathcal{C}}$, allowing impractical configurations to be excluded.

Joint search of policy and sensors. We would like to choose sensor specifications through the task return achievable by optimal policies operating under them, accounting for how behavior can adapt to the available sensing. First, we define the return for arbitrary policies operating under a specification: let Π_σ denote the class of policies that act using only information available under specification σ. For $\pi \in \Pi_\sigma$, we define its expected return in $\mathcal{P}_\sigma$ as

$$J(\pi; \sigma) = \mathbb{E}_{\mathcal{P}_\sigma, \pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]. \tag{2}$$

Then, since the value of a sensor specification is decided by an optimal policy, we can jointly seek an optimal policy and its associated sensor specification subject to some constraints. For a "sensing level" k, let $\Sigma_k = \{\sigma \in \Sigma : |\sigma| = k\}$ denote the admissible specifications containing k sensors:

$$(\pi_k^\star, \sigma_k^\star) \in \underset{\substack{\sigma \in \Sigma_k \\ \pi \in \Pi_\sigma}}{\arg\max}\, J(\pi; \sigma). \tag{3}$$

The solution identifies both an optimal policy at sensing level k and a sensor specification under which it can operate. We seek these solutions across sensing levels, allowing sensing requirements to be traded against task performance. Let $\mathcal{K} \subseteq \{0, \ldots, K\}$ denote the sensing levels of interest for which Σ_k is nonempty. Our desired output is the collection

$$\{(\pi_k^\star, \sigma_k^\star)\}_{k \in \mathcal{K}}. \tag{4}$$

A straightforward approach could be to train many separate policies varying across sensing levels and select the highest-return policy–specification pair at each sensing level. However, repeated training across a combinatorial specification space makes this approach computationally prohibitive.

3 The Sensor Generalist

Rather than doing a computationally expensive joint search over sensor specifications and policies to find the optimal k-sensing level policy for each k, we propose to decompose the problem into two stages. **(1)** A single "sensor generalist" policy is trained under many different sensor specifications of varying sensor levels. **(2)** We search for specifications that achieve high mean return under the trained generalist. Each selected specification is paired with the generalist conditioned on its available observations, yielding a policy–sensor pair without additional policy training; see fig. 2.

Prior work trains shared controllers through pure RL for robustness to sensor failures (Skand et al., 2024) or for morphology search (Fay et al., 2026). Our setting requires learning across varying sensing levels, including highly sparse specifications that can remove both tactile and proprioceptive inputs, resulting in heavy partial observability. Learning effective behavior under these information constraints is challenging, and motivates our design choices, described below.

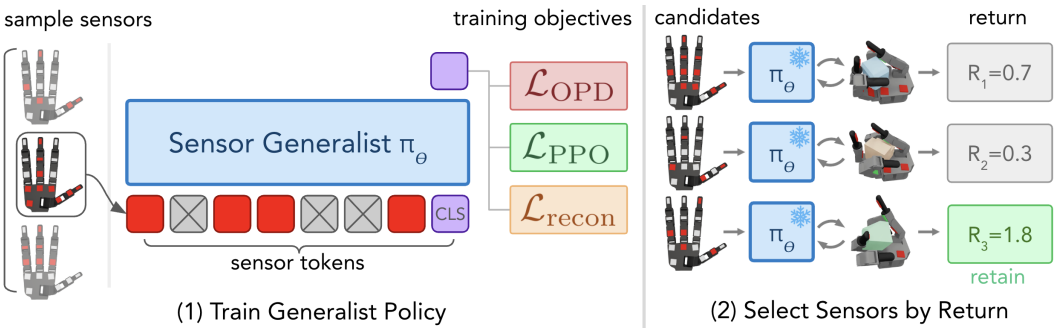


Figure 2: Method overview. Left: the sensor generalist learns to handle arbitrary sensor combinations using a combination of on-policy distillation, RL and reconstruction objectives. Right: once trained, sensors are selected based on their performance under the generalist.

Training. The generalist is trained with three complementary objectives: teacher-student distillation, RL, and auxiliary reconstruction. Distillation provides direct action-level supervision from a pre-trained teacher, reinforcement learning optimizes task return, and reconstruction supports representation learning under partial observations.

We use on-policy distillation (OPD; Agarwal et al., 2024) as the distillation algorithm for the generalist. First, a teacher policy, typically conditioned on the full observation space, $\pi^T(\cdot \mid o^{\mathcal{C}})$ is trained with PPO. The generalist policy π_θ receives only the observations available under the sampled sensor specification σ. Importantly, a different sensor specification is sampled each episode, over the distribution q_{train}, where a sensor level k is first sampled uniformly, and then a specification with k sensors is sampled from the admissible set:

$$k \sim \text{Unif}(\mathcal{K}), \qquad \sigma \sim \text{Unif}(\Sigma_k). \tag{5}$$

For a sampled sensor specification σ, we denote a training trajectory by $\tau^\sigma = \left(o_t^\sigma, o_t^{\mathcal{C}}, a_t, r_t\right)_{t=0}^{H-1}$, where o_t^σ is the observation available to the generalist and $o_t^{\mathcal{C}}$ is the complete observation available to the teacher. At each timestep, the executed action is sampled from the teacher with probability β and from the generalist otherwise, with β gradually annealed toward 0. We denote the resulting trajectory distribution by $\tau^\sigma \sim d^\sigma_{\beta,\theta}$. The distillation objective is defined as the reverse KL divergence between student and teacher policies over trajectories of varying sensor specifications:

$$\mathcal{L}_{\text{OPD}}(\theta) = \mathop{\mathbb{E}}_{\substack{\sigma \sim q_{\text{train}} \\ \tau^\sigma \sim d^\sigma_{\beta,\theta}}} \left[\frac{1}{H} \sum_{t=0}^{H-1} D_{\text{KL}}\left(\pi_\theta(\cdot \mid o_t^\sigma) \,\|\, \pi^T(\cdot \mid o_t^{\mathcal{C}})\right) \right]. \tag{6}$$

The PPO objective $\mathcal{L}_{\text{PPO}}(\theta)$ provides task-based updates alongside teacher supervision, allowing learning to be guided by the returns achieved under each sampled specification. The reconstruction objective $\mathcal{L}_{\text{recon}}(\theta)$ predicts complete observations and privileged simulator information from the generalist's latent representation, encouraging it to encode information about the underlying state that can be inferred from the available inputs. Both objectives are detailed in section A.2. The combined training objective is:

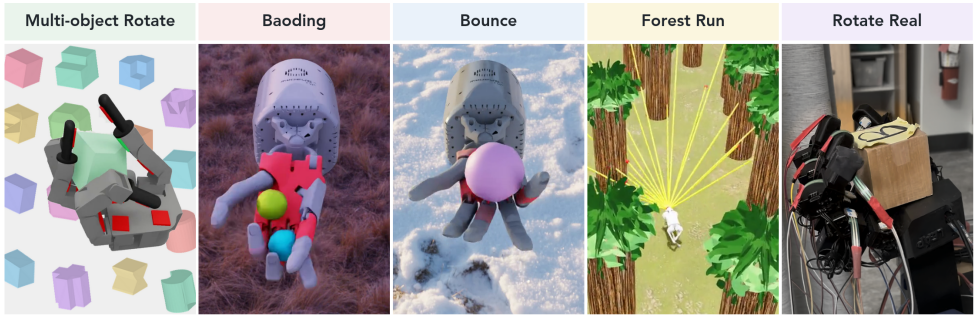
$$\mathcal{L}(\theta) = \lambda_0 \mathcal{L}_{\text{PPO}}(\theta) + \lambda_1 \mathcal{L}_{\text{OPD}}(\theta) + \lambda_2 \mathcal{L}_{\text{recon}}(\theta). \tag{7}$$

Selecting Policy–Sensors Pairs. To select good sensors, we search over specifications that induce high returns with the generalist policy. Many search algorithms are possible such as CEM (Rubinstein, 1999) or evolutionary search; we choose backward selection (Marill & Green, 1963) for its simplicity and small computational overhead. Backward selection iteratively removes sensors that minimally degrade performance. Starting with the current set with k sensors turned on, we drop one sensor out and keep the rest, generating k candidates each with a different sensor removed from the current k sensor set. Then, each candidate is scored by its mean return under the generalist policy,

and the sensor whose removal caused the least degradation is dropped. This is repeated until no sensors are left, generating a specification per sensor level. See section A.3 for details.

Architecture. A standard Transformer encoder is used to process the sensors. Each observation unit, e.g. a tactile observation, joint, or ray, is embedded as a separate token, and a learned `[CLS]` token is prepended to the sequence. Sensors which are not selected are excluded from self-attention in the transformer encoder. We feed only the final `[CLS]` representation to the policy and value heads, see section A.1 for details.

4 Experiments



	Multi-object Rotate	Baoding	Bounce	Forest Run	Rotate Real
Robot	Allegro Hand	Shadow Hand	Shadow Hand	Unitree Go1	LEAP Hand
Sensors	16 tactile	17 tactile, 20 proprio	17 tactile, 20 proprio	20 depth rays	20 tactile
Other inputs	Proprio	None	None	Proprio, last action, base pos	Proprio

Figure 3: Evaluation tasks. Robot, selectable sensors, and always-available inputs are shown below each task; see section D for full environment specifications.

We seek to answer the following questions. **(1)** Does a single generalist policy outperform separately trained specialists at substantially lower computational cost? **(2)** Does the generalist match or exceed specialist performance on identical sensor specifications? **(3)** What mechanisms of the method contribute to its performance? **(4)** What insights do the selected sensors reveal about the task?

We evaluate on five tasks, four in simulation and one in real, summarized in fig. 3. Each is adapted from an established sim-to-real or tactile-sensing benchmark. **Multi-obj. Rotate** adapts the sim-to-real tactile dexterity benchmark of Yin et al. (2023), ported to Isaac Lab; it uses an Allegro hand with 16 tactile sensors measuring contact force magnitude to rotate a set of 16 blocks, cylinders, and irregular objects (fig. 11) along a vertical axis. **Baoding** and **Bounce** are taken directly from RoTO, the Robot Tactile Olympiad benchmark suite (Miller et al., 2025); both use a Shadow Hand with 17 binary tactile sensors, requiring the policy to rotate two Baoding balls or repeatedly bounce a sphere, respectively. The **Forest Run** task is a heavily customized variant of IsaacLab's Go1 locomotion environment. The quadruped must navigate through a $32\,\mathrm{m} \times 32\,\mathrm{m}$ field of procedurally placed trees, using 20 depth rays arranged in arcs originating from the robot's head. Finally, **Rotate Real** evaluates a real-world LEAP dexterous hand with 20 FSR tactile sensors, built on MuJoCo Playground (Zakka et al., 2025), a mature sim-to-real environment library, to rotate various cubes.

4.1 Baselines, Training, and Evaluation

We compare the performance of our method against several baselines (see section B for details), reporting mean episodic return over varying sensor levels.

- **Specialist.** Our specialist baseline represents a standard approach to evaluating candidate sensor configurations: independently training a policy for each configuration using PPO with an asymmetric critic. Up to $\sim$100 distinct PPO policies are trained per task, with randomized configurations and seeds per sensing level, requiring 8–24 GPU days and 14–20 billion steps to train. It serves as a valuable comparison point to evaluate how efficient the sensor generalist is, which is a single policy trained for $\sim$0.1 to 1.2 billion steps.

- **Teacher Backward Selection.** Another strategy is to prune away sensors that minimally degrade the performance of a nominal controller that operates over the full observation space (Melnik et al., 2021). To instantiate this baseline, backward selection is applied to the teacher policy.
- **ℓ_1-norm.** The L1 norm is used to sparsify features in machine learning (Tibshirani, 1996). We use teacher-student distillation to train a student policy with sparse inputs by adding per-sensor sigmoid gates with a L1 penalty. After training, only gates exceeding 0.5 are used. We train three students with different L1 weights, $\{1e-2, 1e-3, 1e-4\}$.
- **State-estimator.** Motivated by approaches that first train an explicit state estimator for a downstream state-conditioned policy (Yu et al., 2017; Ji et al., 2022), we consider a supervised learning approach by training an encoder to reconstruct the full observations from masked ones over the full observation teacher policy distribution. At test time when sensors are missing, the reconstructed observation is used for teacher policy inference.

For the generalist policies for different tasks, we use the same loss weights, optimizer settings, batch sizes, and a DAgger annealing schedule, differing in learning rate (3×10^{-4} for Multi-obj. Rotate and Forest Run; 3×10^{-5} for Baoding and Bounce), LR schedule, and training budget. For Multi-obj. Rotate, Baoding, Bounce, and Forest Run, generalist training uses 200M, 1.2B, 136M, and 200M steps, respectively, while teacher acquisition requires approximately 1B, 0, 1.7B, and 1B additional steps. We found that for Bounce, using a teacher with reduced sensing was better for distillation. See full details are in section A.4.

4.2 Results

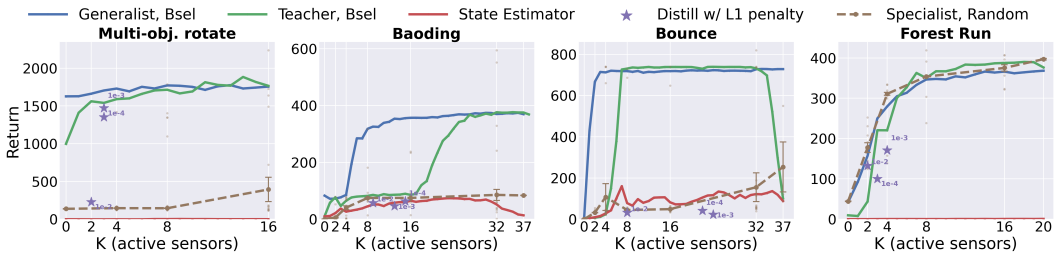


Figure 4: The generalist and its backward selected sensors outperform or match the baselines.

Our method achieves strong returns with substantially less training. In fig. 4, the generalist with backward-selected sensors consistently outperforms or matches the random-specialist baseline across tasks and sensing levels. On Multi-obj. Rotate, it achieves approximately 1300–1600 higher return across sensing levels. On Baoding, its advantage grows sharply as the first few sensors are added, reaching approximately 250–300 higher return for $k \geq 8$. On Bounce, it maintains returns near 720 with as few as four sensors, exceeding the random-specialist baseline by approximately 450–700 for $k \geq 4$. On Forest Run, performance is comparable, with returns within approximately 10% of the random-specialist baseline for $k \geq 8$. These results require substantially fewer training samples: generalist training requires $69\times$, $17\times$, $150\times$, and $101\times$ fewer environment steps than the complete specialist sweeps for the four tasks. Even if we account for teacher training, the training efficiency of our framework is at least $11\times$ higher across tasks. Teacher backward selection is competitive at higher sensing levels but underperforms as sensors are removed. The state-estimator and ℓ_1-regularized baselines also achieve lower returns at comparable sensing levels. Together, these results demonstrate that amortizing policy learning across sensor specifications yields high-performing sensor–policy pairs with substantially less training than independently learning specialists. See the website for videos.

To separate the controller quality from sensor selection, we compare the generalist with specialists conditioned on identical specifications. We evaluate two types of specifications, randomly sampled ones and ones discovered through our backward selection process. In fig. 5, the generalist achieves higher returns than specialists across sensing budgets for both discovered and random specifications in Multi-obj. Rotate, Baoding, and Bounce. On Forest Run, the performance is comparable on discovered specifications, but specialists outperform the generalist on random specifications. Overall, these results show that the gains of the full method arise from having a well-trained generalist policy alongside choosing high performing specifications.

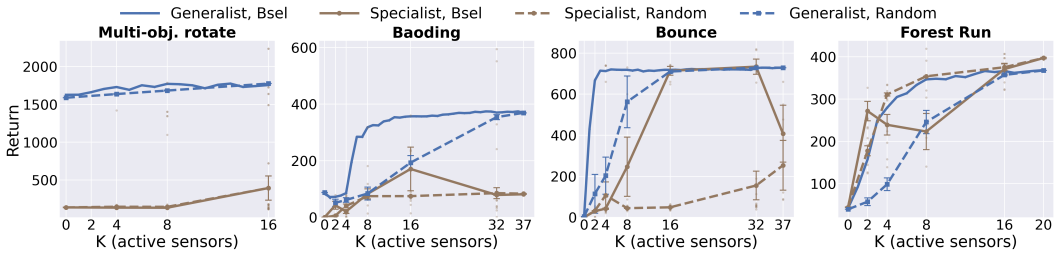


Figure 5: Backward selection improves generalist returns over random sensor selection. On identical specifications, the generalist outperforms specialists in most tasks.

Next, in fig. 5, by comparing the solid and dashed lines of the generalist, we isolate the effect of sensor selection. Backward selection improves average return over random selection on all four tasks, varying from moderate gains of $\sim$ 150 return in Multi-obj. Rotate to substantial gains of $200 - 600$ in Baoding, Bounce, and Forest Run. This validates that the generalist can be used to identify useful sensor configurations without retraining.

4.2.1 Qualitative Analysis of Policies and Sensor Specifications

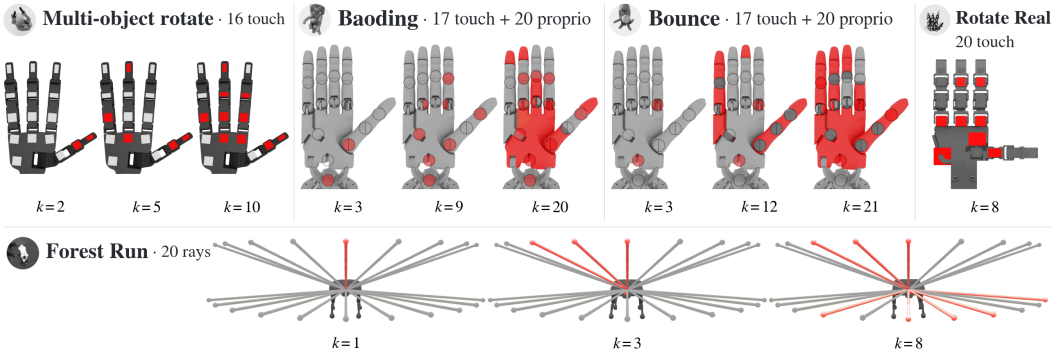


Figure 6: **Selected sensors** across tasks and sensing budgets; selected inputs are red. In Baoding and Bounce, circles denote joint inputs; left / right semicircles indicate rotational / flexion joints.

Across tasks, our method discovers performant policies with non-intuitive, often compact sensor specifications. See fig. 6 for selected visualizations; complete visualizations appear in figs. 13 to 16.

Dexterous manipulation can require surprisingly little tactile sensing. Jointly selecting tactile and proprioceptive inputs in Baoding and Bounce reveals that tactile sensing is not always the most valuable modality, even in contact-rich tasks. See fig. 6 top row, Baoding removes all tactile sensors by nine remaining joint inputs while retaining 90% of full-sensor return. Bounce retains tactile sensors to smaller budgets, yet three joint inputs alone preserve 97% of return. Similarly, Multi-object Rotate retains 88% after removing all 16 tactile sensors while preserving proprioception. Under our evaluated conditions, tactile feedback thus contributes relatively modest gains, and Baoding and Bounce require only sparse proprioceptive inputs to sustain strong performance.

Activation frequency does not consistently predict sensor utility. Although activation frequency is commonly used to gauge sensor importance (Tao et al., 2024), frequently activated sensors are not consistently retained longer in any of our three manipulation tasks. Following fig. 18, in Multi-object Rotate, the thumb-tip sensor is retained last despite activating 2% of the time, while the middle-proximal sensor activates 66% of the time but is removed much earlier. Likewise, Baoding removes its most active sensor, on the proximal index finger, before several less active sensors; Bounce retains several rarely active sensors beyond its more active palm sensor.

Unintuitive configurations can reflect learned control strategies. Forest Run's sparser configurations in fig. 6 favor upper-right range rays, matching the policy's tendency to move diagonally

rightward and placing sensing along its typical path (see fig. 17 for an occupancy visualization). In Bounce, late retention of thumb and ring-finger tactile sensors aligns with using these contacts to trap the ball, while the retained wrist input supports the bouncing motion. Similarly, Multi-object Rotate retains the thumb-tip sensor longest, consistent with the thumb's role in rotation. These observations help explain the selected configurations through the policies' learned strategies, see website for videos.

4.3 Ablations

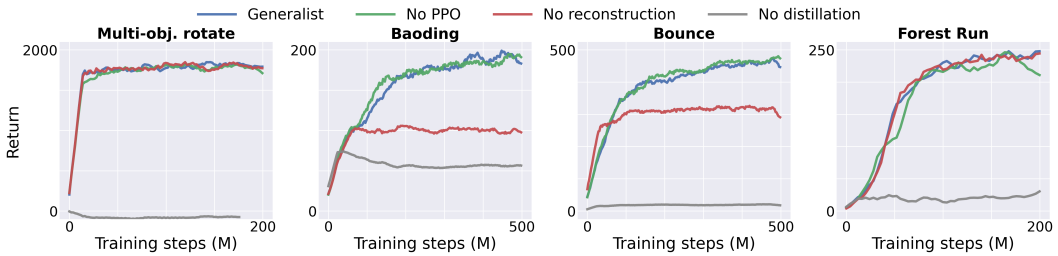


Figure 7: Distillation is essential, and reconstruction is important for Baoding and Bounce.

We ablate the generalist's training components and report mean return over the training distribution of sensor specifications (fig. 7). **No-distillation**: Removing the KL objective and DAgger exploration reduces final returns by approximately 70% in Baoding and roughly 90% or more in the other tasks. We hypothesize that teacher guidance eases exploration and policy optimization under severe partial observability. **No-reconstruction**: Removing reconstruction reduces final returns by roughly 50% in Baoding and 35% in Bounce, with little effect in Multi-obj. Rotate and Forest Run. We hypothesize that reconstruction is particularly useful under the more severe masking in Baoding and Bounce, where proprioceptive inputs can be removed alongside tactile inputs. **No-PPO** has comparable performance in the manipulation tasks but approximately 10–15% lower final return in Forest Run, suggesting that distillation and reconstruction recover most of the full method's performance. Together, these results support teacher guidance and auxiliary reconstruction for training a generalist across the diverse, partially observed control problems encountered during sensor selection.

4.4 Real World Validation

We validate our discovered specifications on a cube rotation task for a LEAP dexterous hand (Shaw et al., 2023) with 20 mounted FSR binary tactile sensors (Yin et al., 2023) in the real world by evaluating the performance of specialist policies trained on our optimized specifications against random specifications, see fig. 3 and section G for pictures and details. The generalist policy is first trained in MuJoCo Playground (Zakka et al., 2025) to rotate objects. Evolutionary search is used to find high performing specifications with $k = 8$ sensors; it is used here because it moderately outperforms backward selection at the cost of increased compute, see section F. The evolved specifications place sensors on the thumb, outer areas of the palm, and on the parts of the finger closest to the palm, see fig. 23. We believe this placement helps the policy localize the cube.

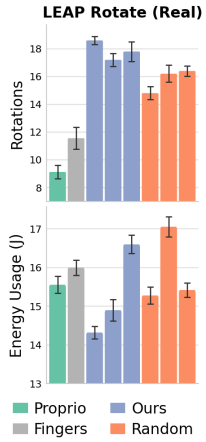


Figure 8: Real robot results.

We train specialist policies on different specifications using MuJoCo Playground's sim-to-real policy training pipeline for LEAP cube rotation. To keep comparisons fair, the policy training pipeline is fixed across policies (same PPO hyperparameters, domain randomization parameters, and training budget), allowing us to isolate the performance difference between policies to their different touch configurations.

We evaluate the sim-to-real performance of policies trained on the following sensor configurations: **Ours**: top 3 sensor specs from evolutionary search. **Random**: 3 random specs with $k = 8$ sensors. Next, we conducted a literature search on tactile sensor placement Lee et al. (2024); Guo et al. (2024); Tang et al. (2025) and found a recurring configuration where tactile sensors are concentrated

on the fingers and sparse on the palm, we call this "**Fingers**". Note this configuration uses 13 sensors, five more than ours or random. **Proprio**: No tactile sensors, relying only on proprioception.

Each policy is evaluated over 15 trials on two metrics, number of cube rotations (90° turns) performed in 25 seconds, and energy usage. As seen in fig. 8, our selected specifications have more rotations and lower energy usage than baselines. We observed that the evolved specification policies exhibited a more deterministic, reactive behavior towards spinning the cube while random and proprioception policies display repetitive, open-loop finger motions with large movements. This often resulted in the cube slipping or getting stuck. The baselines would also move the fingers independent of object pose, resulting in wasted movement. See the website for videos.

5 Other Related Work

Robust Policies under Partial Sensing. A related line of work trains policies to be robust to partial sensing in general, without consideration for sensor selection. Prior approaches distill a teacher with full state information into a partially observed student (Kumar et al., 2021; Ross et al., 2011; Lee et al., 2020) or use a full-state critic (Pinto et al., 2018). Other works train with sensor dropout to encourage robustness to sensor failure (Liu et al., 2017; Skand et al., 2024), while architectures for variable cardinality or masked inputs allow networks to handle a changing subset of inputs (Lee et al., 2019; Jaegle et al., 2021). Most closely related, Skand et al. (2024) train a Transformer policy with input dropout with RL to improve robustness to sensor failures. In contrast, we select sensors across a broad range of sensing levels and modalities, and find that RL is insufficient for training generalist policies in our tasks.

Hardware-Software Co-Design. Also closely related is work in hardware-software co-design, which jointly optimizes morphology and control to produce artificial agents optimal in both body and behavior. These methods span evolutionary approaches for designing simulated creatures (Sims, 1994; Lipson & Pollack, 2000; Cheney et al., 2014), as well as more recent gradient-based (Ha, 2019; Yuan et al., 2022; Matthews et al., 2023; Chen et al., 2021; Xu et al., 2021; Lu et al., 2025) and bi-level optimization techniques (Dai et al., 2026; Luck et al., 2020; Islam et al., 2024). In parallel, advances in cross-embodiment policy generalization (Xiong et al., 2023; Doshi et al., 2024; Huang et al., 2020; Furuta et al., 2023) have enabled shared controllers that operate across diverse morphologies to support co-design (Gupta et al., 2021; Li et al., 2025; Strgar & Kriegman, 2026; Schaff et al., 2019; Dong et al., 2024; Fay et al., 2026). While co-design uses shared controllers to search over morphology, we use them to search over sensors and contribute a generalist policy training framework that addresses severe partial observability unique to our setting.

6 Conclusion

We asked which sensing capabilities are most valuable for the tasks a robot must perform. We formulate this question as a joint search for policies and their associated sensor specifications that maximize task return. Our Sensor Generalist amortizes policy learning across configurations, enabling lightweight search for high-performing policy–sensor pairs at different sensing levels. Our experiments in real and simulated dexterous manipulation and quadrupedal locomotion tasks show that our approach discovers performant policies and sensor specifications. The discovered sensors often run counter to human intuition and reveal surprising insights about the task itself.

Our approach and findings are not without limitations, and there is much future work to be done. First, generalist training currently relies on GPU-accelerated simulation to make training and deployment feasible. Many tasks and sensors do not have readily available simulations, so having an approach that can learn from real-world data would be very important. Next, our tasks are limited to high frequency, low-level sensorimotor tasks (dexterous manipulation, quadruped locomotion), their associated sensors (proprioception, tactile, and rangefinders), and consider only the placement and inclusion of sensors as design choices. Broadening the sensory design space to consider additional axes like resolution, noise, computation, as well as including other common sensors like cameras is left to future work. Finally, our approach selects sensors for narrower tasks, but it would be interesting to see how sensory requirements for a policy change if we consider large task spaces such as in a generalist robotics setting, or within phases of a singular task. We hope our work offers a foundation for future avenues of research in these intriguing directions.

REPRODUCIBILITY STATEMENT

We will release all code on the website to ensure reproducibility. Comprehensive details on the environment, method, baselines, training, and hyperparameters are in the Appendix.

AI DISCLOSURE

In this work, we used generative AI tools for assisting with editing and drafting writing, and figure creation. We have used generative AI tools for retrieval and discovery of related work. Any papers suggested by AI were carefully verified by the authors before citing, the authors never directly used AI to generate citations, bibtex, etc. We have used AI for ideation and execution, such as running experiments on the cluster. Experimental results are checked by the authors. We have not used AI to generate synthetic datasets, nor prove mathematical claims. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, volume 2024, pp. 21246–21263, 2024.

Natalie L. Brace and Kristi A. Morgansen. Observability-based sensor selection for a planar bending beam attached to a rotating body. In *2024 American Control Conference (ACC)*, pp. 3972–3979, 2024. doi: 10.23919/ACC60939.2024.10644489.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Tianjian Chen, Zhanpeng He, and Matei Ciocarlie. Co-designing hardware and control for robot hands. *Science Robotics*, 6(54):eabg2133, 2021.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1): 11–23, 2014.

Yanning Dai, Yuhui Wang, Dylan R. Ashley, and Jürgen Schmidhuber. Efficient morphology–control co-design via stackelberg PPO. In *The Fourteenth International Conference on Learning Representations*, 2026.

Heng Dong, Junyu Zhang, and Chongjie Zhang. Leveraging hyperbolic embeddings for coarse-to-fine robot design. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=q9jQPA6zPK.

Ria Doshi, Homer Rich Walke, Oier Mees, Sudeep Dasari, and Sergey Levine. Scaling cross-embodied learning: One policy for manipulation, navigation, locomotion and aviation. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=AuJnXGq3AL.

Kehlani Fay, Darin Djapri, Anya Zorin, James Clinton, Ali El Lahib, Hao Su, Michael Tolley, Sha Yi, and Xiaolong Wang. House of dextra: Cross-embodied co-design for dexterous hands. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 14889–14912, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/18c54ed6e0cc390d750f64927dbc4e93-Paper-Conference.pdf.

Hiroki Furuta, Yusuke Iwasawa, Yutaka Matsuo, and Shixiang Shane Gu. A system for morphology-task generalization via unified representation and behavior distillation. In *International Conference on Learning Representations*, 2023.

Quanquan Gu, Zhenhui Li, and Jiawei Han. Generalized fisher score for feature selection. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, pp. 266–273, Arlington, Virginia, USA, 2011. AUAI Press. ISBN 9780974903972.

Haoran Guo, Haoyang Wang, Zhengxiong Li, He Bai, and Lingfeng Tao. Learning with less: Optimizing tactile sensor configurations for dexterous manipulation. *arXiv preprint arXiv:2409.20473*, 2024.

Agrim Gupta, Silvio Savarese, Surya Ganguli, and Li Fei-Fei. Embodied intelligence via learning and evolution. *Nature Communications*, 12(1), October 2021. ISSN 2041-1723. doi: 10.1038/s41467-021-25874-z. URL http://dx.doi.org/10.1038/s41467-021-25874-z.

David Ha. Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365, 2019.

Hirotaka Hachiya and Masashi Sugiyama. Feature selection for reinforcement learning: Evaluating implicit state-reward dependency via conditional mutual information. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 474–489. Springer, 2010.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4455–4464. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/huang20d.html.

Sharfin Islam, Zhanpeng He, and Matei T. Ciocarlie. Task-based design and policy co-optimization for tendon-driven underactuated kinematic chains. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2024, Abu Dhabi, United Arab Emirates, October 14-18, 2024*, pp. 12016–12023. IEEE, 2024. doi: 10.1109/IROS58592.2024.10802329. URL https://doi.org/10.1109/IROS58592.2024.10802329.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4651–4664. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/jaegle21a.html.

Gwanghyeon Ji, Juhyeok Mun, Hyeongjun Kim, and Jemin Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637, 2022.

Siddharth Joshi and Stephen Boyd. Sensor selection via convex optimization. *IEEE Transactions on Signal Processing*, 57(2):451–462, 2008.

Joanne Taery Kim and Sehoon Ha. Observation space matters: Benchmark and optimization algorithm. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1527–1534. IEEE, 2021.

Yasutaka Kishima and Kentarou Kurashige. Reduction of state space in reinforcement learning by sensor selection. *Artificial Life and Robotics*, 18(1):7–14, 2013.

J Zico Kolter and Andrew Y Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 521–528, 2009.

Arjun Krishna, Edward S Hu, and Dinesh Jayaraman. The value of sensory information to a robot. In *The Thirteenth International Conference on Learning Representations*, 2025.

Mark Kroon and Shimon Whiteson. Automatic feature selection for model-based reinforcement learning in factored mdps. In *2009 International Conference on Machine Learning and Applications*, pp. 324–330. IEEE, 2009.

Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), October 2020. ISSN 2470-9476. doi: 10.1126/scirobotics.abc5986. URL http://dx.doi.org/10.1126/scirobotics.abc5986.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3744–3753. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/lee19d.html.

Kang-Won Lee, Yuzhe Qin, Xiaolong Wang, and Soo-Chul Lim. Dextouch: Learning to seek and manipulate objects with tactile dexterity. *IEEE Robotics and Automation Letters*, 9(12), 2024. ISSN 2377-3774. doi: 10.1109/lra.2024.3478571. URL http://dx.doi.org/10.1109/LRA.2024.3478571.

Muhan Li, Lingji Kong, and Sam Kriegman. Generating freeform endoskeletal robots. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 41952–41976, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/675e371eeeea99551ce47797ed6ed33e-Paper-Conference.pdf.

Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

Guan-Horng Liu, Avinash Siravuru, Sai Prabhakar, Manuela Veloso, and George Kantor. Learning end-to-end multimodal sensor policies for autonomous navigation. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg (eds.), *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pp. 249–261. PMLR, 13–15 Nov 2017. URL https://proceedings.mlr.press/v78/liu17a.html.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Haofei Lu, Zhe Wu, Junliang Xing, Jianshu Li, Ruoyu Li, Zhe Li, and Yuanchun Shi. Bodygen: Advancing towards efficient embodiment co-design. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=cTR17xl89h.

Kevin Sebastian Luck, Heni Ben Amor, and Roberto Calandra. Data-efficient co-adaptation of morphology and behaviour with deep reinforcement learning. In *Conference on Robot Learning*, pp. 854–869. PMLR, 2020.

Yecheng Jason Ma, William Liang, Hungju Wang, Sam Wang, Yuke Zhu, Linxi Fan, Osbert Bastani, and Dinesh Jayaraman. DrEureka: Language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)*, 2024.

Thomas Marill and Donald Green. On the effectiveness of receptors in recognition systems. *IEEE transactions on Information Theory*, 9(1):11–17, 1963.

David Matthews, Andrew Spielberg, Daniela Rus, Sam Kriegman, and Josh Bongard. Efficient automatic design of robots. *Proceedings of the National Academy of Sciences*, 120(41):e2305180120, 2023.

Andrew Melnik, Luca Lach, Matthias Plappert, Timo Korthals, Robert Haschke, and Helge Ritter. Using tactile sensing to improve the sample efficiency and performance of deep deterministic policy gradients for simulated in-hand manipulation tasks. *Frontiers in Robotics and AI*, Volume 8 - 2021, 2021. ISSN 2296-9144. doi: 10.3389/frobt.2021.538773. URL https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.538773.

Elle Miller, Trevor McInroe, David Abel, Oisin Mac Aodha, and Sethu Vijayakumar. Enhancing tactile-based reinforcement learning for robotic control. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Toy96yYopR.

Jun Morimoto, Sang-Ho Hyon, Christopher G Atkeson, and Gordon Cheng. Low-dimensional feature extraction for humanoid locomotion using kernel dimension reduction. In *ICRA*, pp. 2711–2716, 2008.

Trung Nguyen, Zhuoru Li, Tomi Silander, and Tze Yun Leong. Online feature selection for model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 498–506. PMLR, 2013.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 752–759, 2008.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *RSS*, 2018.

Marko Robnik-Šikonja and Igor Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Machine learning*, 53(1):23–69, 2003.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Reuven Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

Charles Schaff, David Yunis, Ayan Chakrabarti, and Matthew R. Walter. Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9798–9805, 2019. doi: 10.1109/ICRA.2019.8793537.

Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand: Low-cost, efficient, and anthropomorphic hand for robot learning. *Robotics: Science and Systems (RSS)*, 2023.

Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

Skand Skand, Bikram Pandit, Chanho Kim, Li Fuxin, and Stefan Lee. Simple masked training strategies yield control policies that are robust to sensor failure. In *8th Annual Conference on Robot Learning*, 2024.

Luke Strgar and Sam Kriegman. Accelerated co-design of robots through morphological pretraining. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2026. URL https://openreview.net/forum?id=WVliGyFwZv.

Min Tang, Xiaoyu Liu, Xiaofeng Qiao, Yuanjie Zhu, Linyuan Fan, Songjun Du, Duo Chen, Jinghui Wang, Zhiyang Zhang, Wanxin Zhang, et al. An optimization strategy allowing a tactile glove with minimal tactile sensors for soft objects identification. *IEEE Journal of Biomedical and Health Informatics*, 2025.

Lingfeng Tao, Jiucai Zhang, Qiaojie Zheng, and Xiaoli Zhang. Curriculum-based sensing reduction in simulation to real-world transfer for in-hand manipulation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6530–6536. IEEE, 2024.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

Jiawen Wei, Fangyuan Wang, Wanxin Zeng, Wenwei Lin, and Ning Gui. An embedded feature selection framework for control. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1979–1988, 2022.

Zheng Xiong, Jacob Beck, and Shimon Whiteson. Universal morphology control via contextual modulation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38286–38300. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/xiong23a.html.

Jie Xu, Tao Chen, Lara Zlokapa, Michael Foshey, Wojciech Matusik, Shinjiro Sueda, and Pulkit Agrawal. An End-to-End Differentiable Framework for Contact-Aware Robot Design. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi: 10.15607/RSS.2021.XVII.008.

Zhao-Heng Yin, Binghao Huang, Yuzhe Qin, Qifeng Chen, and Xiaolong Wang. Rotating without seeing: Towards in-hand dexterity through touch. *Robotics: Science and Systems*, 2023.

Wenhao Yu, Jie Tan, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. In Nancy M. Amato, Siddhartha S. Srinivasa, Nora Ayanian, and Scott Kuindersma (eds.), *Robotics: Science and Systems XIII, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 12-16, 2017*, 2017. doi: 10.15607/RSS.2017.XIII.048. URL http://www.roboticsproceedings.org/rss13/p48.html.

Ye Yuan, Yuda Song, Zhengyi Luo, Wen Sun, and Kris Kitani. Transform2act: Learning a transform-and-control policy for efficient agent design. In *International Conference on Learning Representations*, 2022.

Kevin Zakka, Baruch Tabanpour, Qiayuan Liao, Mustafa Haiderbhai, Samuel Holt, Jing Yuan Luo, Arthur Allshire, Erik Frey, Koushil Sreenath, Lueder A. Kahrs, Carmelo Sferrazza, Yuval Tassa, and Pieter Abbeel. Mujoco playground, 2025. URL https://arxiv.org/abs/2502.08844.

A METHOD DETAILS

A.1 TRANSFORMER ARCHITECTURE AND TOKENIZATION

Every sensor unit is embedded as a separate token, a learned `[CLS]` token is prepended, and a single standard Transformer encoder processes all tokens jointly. Masked-out sensors are excluded from self-attention via a key-padding mask, so the CLS token aggregates only the information from active sensors. The CLS output is passed through a two-layer fusion MLP to produce the policy input z. The token construction varies by task to match each robot's observation space.

A.2 GENERALIST TRAINING OBJECTIVES

OPD objective. For a sampled sensor specification σ, we denote a training trajectory by $\tau^\sigma = \left(o_t^\sigma, o_t^{\mathcal{C}}, a_t, r_t\right)_{t=0}^{H-1}$, where o_t^σ is the observation available to the generalist and $o_t^{\mathcal{C}}$ is the complete observation available to the teacher. At each timestep, the executed action is sampled from the teacher with probability β and from the generalist otherwise, with β gradually annealed to 0. We denote the resulting trajectory distribution by $\tau^\sigma \sim d_{\beta,\theta}^\sigma$. The distillation objective is:

$$\mathcal{L}_{\mathrm{OPD}}(\theta) = \mathop{\mathbb{E}}_{\substack{\sigma\sim q_{\mathrm{train}} \\ \tau^\sigma \sim d_{\beta,\theta}^\sigma}} \left[\frac{1}{H} \sum_{t=0}^{H-1} D_{\mathrm{KL}}\left(\pi_\theta(\cdot \mid o_t^\sigma) \,\|\, \pi^T(\cdot \mid o_t^{\mathcal{C}})\right) \right]. \tag{8}$$

DAgger schedule. At each timestep, each training environment independently executes the teacher with probability $\beta(t)$ and the student otherwise (a per-environment Bernoulli draw). The mixing coefficient follows an exponential decay:

$$\beta(t) = \max\left(\beta_{\mathrm{end}},\ \beta_{\mathrm{start}} \cdot \exp\left(-\frac{k\,t}{T}\right)\right), \tag{9}$$

with β_{start}=1, β_{end}=0, decay constant k=3, and T=100M environment steps across all tasks. At t=T the teacher fraction is $e^{-3} \approx 5\%$, so the generalist is trained almost entirely on its own trajectory distribution after T steps. The teacher encoder and policy are frozen throughout; the student's observation normalizer is seeded from the teacher checkpoint at initialization to prevent distribution shift in the early rollouts.

PPO objective. The PPO objective is also computed over $\tau^\sigma \sim d_{\beta,\theta}^\sigma$. Let $z_t = f_\theta(o_t^\sigma)$ be the CLS latent, $\hat{A}_t$ the GAE advantage estimate with γ=0.99, λ=0.95, and $r_t(\theta) = \pi_\theta(a_t \mid o_t^\sigma)/\pi_{\theta_{\mathrm{old}}}(a_t \mid o_t^\sigma)$ the probability ratio against the previous policy iterate. The policy and value losses are:

$$\mathcal{L}_{\mathrm{clip}}(\theta) = -\mathop{\mathbb{E}}_{\substack{\sigma\sim q_{\mathrm{train}} \\ \tau^\sigma \sim d_{\beta,\theta}^\sigma}} \left[\frac{1}{H} \sum_{t=0}^{H-1} \min\!\big(r_t(\theta)\,\hat{A}_t,\ \mathrm{clip}(r_t(\theta),\,1{-}\epsilon,\,1{+}\epsilon)\,\hat{A}_t\big) \right], \tag{10}$$

$$\mathcal{L}_V(\phi) = \mathop{\mathbb{E}}_{\substack{\sigma\sim q_{\mathrm{train}} \\ \tau^\sigma \sim d_{\beta,\theta}^\sigma}} \left[\frac{1}{H} \sum_{t=0}^{H-1} \left(V_\phi(z_t,\, g_t) - \hat{R}_t\right)^2 \right], \tag{11}$$

with clipping coefficient ϵ=0.2 and discounted-return target $\hat{R}_t$. The combined PPO objective is $\mathcal{L}_{\mathrm{PPO}}(\theta,\phi) = \mathcal{L}_{\mathrm{clip}}(\theta) + c_v\,\mathcal{L}_V(\phi)$ with c_v=1.0; no entropy bonus is used. The value network V_ϕ takes as input the actor latent z_t concatenated with the privileged state g_t which is available from the simulator but never passed to the actor, following the asymmetric actor-critic design (Pinto et al., 2018). For teacher-collected steps, $\mathcal{L}_{\mathrm{clip}}$ is zeroed on those transitions; $\mathcal{L}_V$, $\mathcal{L}_{\mathrm{recon}}$, and $\mathcal{L}_{\mathrm{KL}}$ still receive gradients from all transitions.

Reconstruction objective. The reconstruction objective $\mathcal{L}_{\mathrm{recon}}$ trains two lightweight MLP decoders on the generalist's CLS latent $z_t = f_\theta(o_t^\sigma)$. Both teacher- and student-collected transitions contribute to the gradient; the decoder targets are privileged quantities collected in the rollout buffer but never passed to the actor.

Privileged-state reconstruction. A two-layer MLP decoder d_g predicts a task-specific privileged state vector g_t from the CLS latent:

$$\mathcal{L}_g(\theta) = \mathop{\mathbb{E}}_{\substack{\sigma\sim q_{\text{train}} \\ \tau^\sigma \sim d^\sigma_{\beta,\theta}}} \left[\frac{1}{H} \sum_{t=0}^{H-1} \| d_g(z_t) - g_t \|_2^2 \right]. \tag{12}$$

The target g_t is task-dependent: ball positions and velocities for Baoding and Bounce, object pose and velocity for Multi-obj. Rotate, and unmasked ray distances plus foot contacts for Forest Run.

Observation reconstruction. A second two-layer MLP decoder d_{obs} reconstructs the *unmasked* proprioceptive and tactile observations from the masked latent. Let $p_t^\star$ and $\tau_t^\star$ denote the unmasked proprioception and tactile readings, respectively:

$$\mathcal{L}_{\text{obs}}(\theta) = \mathop{\mathbb{E}}_{\substack{\sigma\sim q_{\text{train}} \\ \tau^\sigma \sim d^\sigma_{\beta,\theta}}} \left[\frac{1}{H} \sum_{t=0}^{H-1} \Big(\text{MSE}(d_p(z_t),\, p_t^\star) + \mathcal{L}_\tau(d_\tau(z_t),\, \tau_t^\star) \Big) \right], \tag{13}$$

where $\mathcal{L}_\tau$ is a per-sensor class-weighted binary cross-entropy for tasks with binary contact readings (Baoding, Bounce) and MSE for tasks with continuous magnitudes (Multi-obj. Rotate, Forest Run).

The combined reconstruction loss is:

$$\mathcal{L}_{\text{recon}}(\theta) = \mathcal{L}_g(\theta) + \mu_{\text{obs}}\, \mathcal{L}_{\text{obs}}(\theta), \tag{14}$$

with μ_{obs}=0 for Baoding and Bounce and μ_{obs}=0.01 for Multi-obj. Rotate and Forest Run. All decoder parameters are optimized jointly with the encoder.

A.3 Backward Selection

Starting from the full sensor set $\sigma^{(K)} = \mathcal{C}$, backward selection iteratively removes the sensor whose removal least degrades performance under the frozen generalist, producing one specification per sensing level $k \in \{0, \ldots, K\}$.

At step k (removing from the current k-sensor specification $\sigma^{(k)}$), k candidate specifications are formed, each with a different active sensor removed. A "keep-current" candidate (the specification unchanged) is added, giving $k+1$ candidates total. Each candidate is scored by estimating its mean episodic return $\widehat{J}(\sigma)$ over N rollout episodes, using E parallel environments to collect episodes efficiently. The sensor whose removal yields the highest estimated return is permanently dropped:

$$\sigma^{(k-1)} = \sigma^{(k)} \setminus \left\{ \mathop{\arg\max}_{c \in \sigma^{(k)}} \widehat{J}\Big(\sigma^{(k)} \setminus \{c\}\Big) \right\}. \tag{15}$$

The keep-current candidate provides an unbiased re-evaluation of $\sigma^{(k)}$ (reported as the score for sensing level k in the results). The full procedure runs in $K+1$ evaluation steps and requires no policy updates. The hyperparameters are N (episodes per candidate) and E (parallel environments); per-task values are given in Table 5.

A.4 Training the Generalist

A.4.1 Tokenization

We describe each variant below; hyperparameters are summarized in Table 1.

Shadow Hand (Baoding, Bounce). Each of the 20 joints is embedded as one token, constructed from the joint's position, velocity, position error, and last action stacked across the S=4 observation frames (4 features $\times$ S frames = 16 values $\to$ d). Each of the 17 tactile sensors is a separate token formed from its binary contact reading across S frames (S values $\to$ d). Token layout: $[\texttt{CLS},\ \text{joint}_{0..19},\ \text{tactile}_{0..16}]$, for 38 total tokens.

Allegro Hand (Multi-obj. Rotate). Each of the 16 joints is a token formed from position and PD target across S=4 frames (2 $\times$ S=8 values $\to$ d). The 3-dimensional rotation-axis goal is split into 3 always-on tokens (1 scalar $\times$ S frames each), representing the per-episode task specification. Each of the 16 tactile sensors is a token formed from its force-magnitude reading across S frames. Token layout: $[\texttt{CLS},\ \text{joint}_{0..15},\ \text{goal}_{0..2},\ \text{tactile}_{0..15}]$, for 36 total tokens.

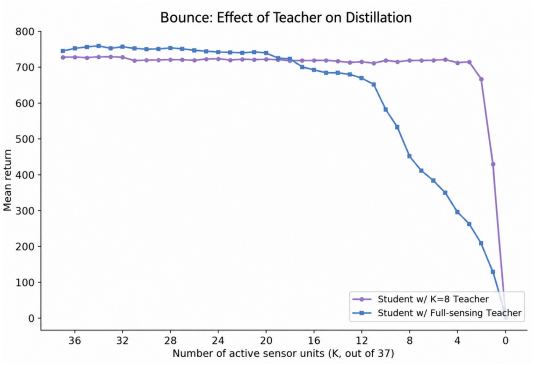


Figure 9: In Bounce, we compare a student trained with all sensor teacher versus a student trained with a teacher that only sees 8 sensors.

Unitree Go1 (Forest Run). Each of the 12 leg joints is a token formed from position, velocity, and action across S=1 frames (3 features $\rightarrow d$). The 10-dimensional body-state vector (linear velocity, angular velocity, projected gravity, and yaw) is always-on and embedded as a single base token. Each of the 20 LiDAR rays is a maskable token (1 reading $\times$ S frames $\rightarrow d$). Token layout: $[\texttt{CLS},\ \text{joint}_{0..11},\ \text{base},\ \text{ray}_{0..19}]$, for 34 total tokens.

Table 1: Encoder hyperparameters per task. All use a standard Transformer encoder (pre-norm, ReLU feed-forward, $4\times$ FFN expansion, no dropout) with a two-layer fusion MLP after the CLS token. S = observation stack depth.

Task	Encoder variant	d	Heads	Layers	Tokens	S	Fusion MLP	z dim
Baoding	Shadow	128	8	4	38	4	[256, 128]	128
Bounce	Shadow	128	8	4	38	4	[256, 128]	128
Multi-obj. Rotate	Allegro	256	16	6	36	4	[1024, 512]	512
Forest Run	Go1	256	16	6	34	1	[1024, 512]	512

A.4.2 Acquiring Teacher Policies

Over all tasks, the cost of teacher acquisition ranged from doing no training, by using a preexisting checkpoint, up to 1.7B steps. This is still much cheaper compared to $\sim$14-20B steps used for the exhaustive random specialist baselines.

Teacher policies are trained via PPO. We default to full-sensing teachers ($K = K_{\max}$), which are the easiest to train; for some tasks a pretrained checkpoint is used directly. We hypothesized that there may be gains by using a reduced-sensor teacher since that narrows the imitation gap. Because reduced sensor teachers are harder to train since we need to decide what the reduced observation space is and deal with partial observability, we designed an efficient iterative search to find teacher policies with lower sensor levels.

We start with sensor levels to consider, e.g. $\mathcal{K} = (32, 16, 8, 4)$. At each level K_i, we sample 5 random sensor specifications and train a policy up to $T_{\max} = 200$M steps; trials are pruned at $T_{\text{prune}} = 100$M steps if performance is flat. The first trial exceeding a task-specific success threshold is taken as the teacher for that level; and we move on to the next level. If all trials are pruned, the search terminates. For the Bounce task, the search found a reduced-sensor teacher with 8 sensors that did as well as the full sensing teacher, so we used the reduced sensor teacher, see 9. In the other tasks, we found that it did not find a successful reduced-sensing teacher within the allotted search budget, or we found that using the pre-existing checkpoint was sufficient.

A.4.3 Training Details

Rollouts are collected with E_{train}=4096 parallel environments for 16 steps per update, yielding a rollout buffer of $4096 \times 16 = 65{,}536$ transitions. Each update performs 4 learning epochs over

Task	Acquisition Strategy	Sensing level	Cost
Multi-obj. Rotate	PPO, 5 seeds	Full (K=16)	∼1B steps
Baoding	Preexisting checkpoint	Full (K=37)	0 steps
Bounce	Cascaded search	Reduced (K=8)	∼1.7B steps
Forest Run	PPO, 5 seeds	Full (K=20)	∼1B steps

Table 2: Teacher policy acquisition per task.

8 mini-batches with the Adam optimizer. The PPO, OPD, and GT-reconstruction loss weights are equal (λ_0=λ_1=1.0, $\lambda_{2,g}$=1.0) across all tasks. The DAgger mixing coefficient decays from β_{start}=1 towards β_{end}=0 over T=100M environment steps in all tasks. All tasks train the GT-state reconstruction decoder on a task-specific privileged target g_t ($\lambda_{2,g}$=1.0). Tasks further differ in learning rate, observation stack depth, and the observation-reconstruction weight μ_{obs}; per-task settings are in Table 4. Complete shared hyperparameters are in Table 3.

Table 3: Shared generalist training hyperparameters (identical across all four tasks).

Hyperparameter	**Value**
Parallel training environments E_{train}	4096
Rollout length	16 steps
Learning epochs per update	4
Mini-batches per epoch	8
Discount factor γ	0.99
GAE λ	0.95
PPO clip ϵ	0.2
Value clip	0.2
Gradient norm clip	1.0
Entropy bonus	none
PPO weight λ_0	1.0
OPD weight λ_1	1.0
GT-recon weight $\lambda_{2,g}$	1.0
DAgger β_{start}	1.0
DAgger β_{end}	0.0
DAgger decay horizon T	100M steps

Table 4: Per-task generalist training settings.

	Multi-obj. Rotate	**Baoding**	**Bounce**	**Forest Run**
Learning rate	3×10^{-4}	3×10^{-5}	3×10^{-5}	3×10^{-4}
LR schedule	constant	cosine decay	cosine decay	constant
Obs stack S	4	4	4	1
Obs-recon weight μ_{obs}	0.01	0.0	0.0	0.01
GT recon target g_t	object pose + vel	ball pos + vel	ball pos + vel	rays + foot contacts
Training steps	200M	1,200M	136M	200M

Backward selection details. Table 5 lists the backward selection configuration per task. For Baoding and Bounce, the script allocates $E/(k+1)$ environments per candidate at each removal step k and runs one episode per environment, so the number of episodes per candidate rises from ≈89 (first step, k=37 candidates) to 1700 (last step, k=1 candidate remaining). For Multi-obj. Rotate and Forest Run, the script accumulates episodes across multiple rollout passes until a fixed target is reached.

Table 5: Backward selection settings per task.

Task	**Envs** E	**Episodes / candidate**
Multi-obj. Rotate	3200	5500 (fixed target)
Baoding	3400	$\lfloor E/(k+1) \rfloor$ at step k
Bounce	3400	$\lfloor E/(k+1) \rfloor$ at step k
Forest Run	3200	500 (fixed target)

B Baseline Details

B.1 Specialist Baseline Training

For each task, our specialist baselines train a PPO policy from scratch on a *fixed* sensor mask exposing only K of the task's sensor units, with no distillation or auxiliary objective. Masks are drawn two ways: (i) uniformly at random among the $\binom{N}{K}$ possible K-of-N subsets (the *random-mask* baseline), and (ii) via greedy backward selection on the task's generalist policy (the *bsel* baseline). Every specialist, across all four tasks, is trained with the same architecture/PPO hyperparameters as the corresponding task's teacher, for a fixed budget of 200M environment steps.

Multi-obj. Rotate. Specialists were trained at $K \in \{0, 4, 8, 16\}$ (of 16 tactile sensor units), 5 seeds per mask. $K = 0$ and $K = 16$ admit only one possible mask (5 seeds and 14 seeds respectively, the latter widened to account for high seed-to-seed variance); $K = 4$ and $K = 8$ each use 5 distinct masks (4 random plus the bsel-path mask, reused within the same sweep rather than trained separately), 5 seeds each. This gives 69 specialist runs.

Baoding and Bounce. Both tasks share the same 37-unit coupled sensor space and were swept at $K \in \{0, 2, 4, 8, 16, 32, 37\}$. The random-mask baseline uses 5 distinct masks per K (1 at $K = 0/37$, which admit only one mask) $\times$ 3 seeds (81 runs); the bsel-path baseline adds one greedily-selected mask per $K \times 3$ seeds (21 runs), for 102 specialist runs per task.

Forest Run. Specialists were swept at $K \in \{0, 2, 4, 8, 16, 20\}$ (of 20 sensor units). The random-mask baseline uses 8 distinct masks per K at $K \in \{2, 4, 8, 16\}$ ($\times$ 1 seed) plus 3 seeds at the single mask possible at $K = 0/20$ (38 runs); the bsel-path baseline uses one mask per K from 0 to 20×3 seeds (63 runs), for 101 specialist runs.

Table 6 summarizes the total number of specialist policies trained and the resulting training budget for each task.

Table 6: Specialist baseline training budget per task. Every run uses the same 200M-environment-step budget.

Task	Specialist policies trained	Total training samples
Multi-obj. Rotate	69	13.8B
Baoding	102	20.4B
Bounce	102	20.4B
Forest Run	101	20.2B
Total	374	74.8B

B.2 ℓ_1-Gating

The ℓ_1-sparsity gate uses a learnable per-sensor gate and a KL distillation loss from a frozen teacher. Each sensor unit i is assigned a scalar logit $\gamma_i \in \mathbb{R}$. Before every rollout the logits are binarized at threshold 0.5:

$$m_i = \mathbf{1}[\sigma(\gamma_i) > 0.5],$$

and m is written to the environment's sensor mask so that training and evaluation use identical hard-masked observations with no distribution shift. Gates are initialized at $+0.1$ (sigmoid ≈ 0.525, all sensors active at the start of training).

Loss and gradient path. The full training objective is

$$\mathcal{L} = \mathcal{L}_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{teacher}}) + \lambda \textstyle\sum_i \sigma(\gamma_i).$$

The L1 term pushes all gate logits downward; the distillation term must push useful gates back up. Because hard thresholding is not differentiable, the KL gradient reaches the gate logits through a **straight-through estimator (STE)**: during the loss computation the sensor values are multiplied by $\mathrm{STE}_i = \mathbf{1}[\sigma(\gamma_i){>}0.5] + (\sigma(\gamma_i) - \sigma(\gamma_i).\mathrm{detach}())$, which is binary in the forward pass (matching rollout behavior) but passes the full sigmoid gradient in the backward pass. Additionally, the transformer's *attention mask* is set to all-ones during the loss computation so that even sensors whose STE gate evaluates to zero can still receive KL gradient and recover; without this, an inactive gate would see zero gradient from the distillation objective and could never revive. The L1 penalty is computed on the soft sigmoid values $\sigma(\gamma_i)$ (not the STE), ensuring it applies uniformly to all gates regardless of their current binary state.

We sweep $\lambda \in \{10^{-4}, 10^{-3}, 10^{-2}\}$. Training runs for 200 M environment steps with the same encoder architecture and PPO hyperparameters as the generalist. Each run produces a single binarized mask at convergence.

B.3 Teacher Backward Selection

Greedy backward selection can be applied directly to the *teacher* policy—typically the fully-observed specialist—without training any student. At each step, we zero-fill each remaining sensor in turn and evaluate the resulting $K-1$-sensor teacher policy; the sensor whose removal causes the smallest performance drop is permanently removed.

Because the teacher is often trained with all sensors always active, it is not robust to missing sensors, and its performance degrades faster than the generalist's as K decreases.

B.4 State-Estimator Baseline

The state-estimator baseline trains an encoder purely on a reconstruction objective: given a randomly masked observation, reconstruct the full unmasked proprioception and tactile readings.

Architecture. The encoder is the same transformer used by the generalist. A two-layer MLP decoder maps the CLS latent $\mathbf{z} \in \mathbb{R}^{d_z}$ ($d_z = 128$ for Baoding and Bounce, 512 for Multi-obj. Rotate and Forest Run) to a concatenated reconstruction $[\hat{p}_{\mathrm{prop}} \,\|\, \hat{t}_{\mathrm{tac}}]$.

All policy-gradient and value terms are disabled; only the reconstruction loss and the encoder optimizer are active. The environment is driven by the frozen teacher throughout to ensure training covers a useful state distribution. Each run trains for 50 M steps.

Inference. The trained decoder is kept at evaluation time. The reconstruction feeds the teacher's *own* encoder–policy pipeline.

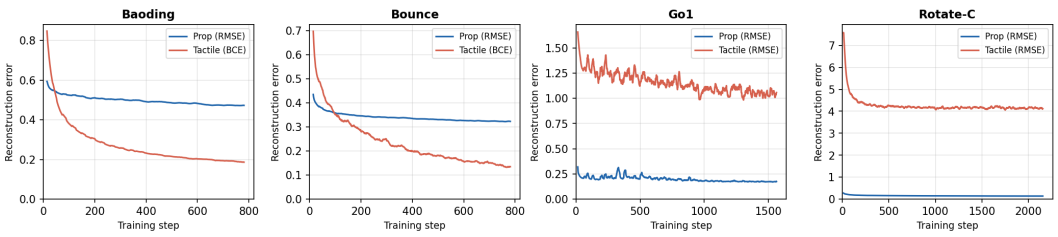


Figure 10: Training curves of the state estimator.

Reconstruction quality. Training losses do converge: the reconstruction curves (Figure 10) show the decoder fitting the training distribution. However, the reconstruction is not accurate enough for the teacher: even modest residual errors in the reconstructed observations are sufficient to destabilize a teacher policy that was trained on precise, unperturbed inputs. This gap between reconstruction accuracy and the accuracy required for downstream control is the core limitation of the approach.

C Evaluation Details

All reported results in simulation are means with $\pm$ standard error ($\hat{\sigma}/\sqrt{n}$) computed over multiple policies (aggregated over both seeds and sensor specifications) for the specialist baselines in figs. 4 and 5 and multiple masks for the random-mask generalist in fig. 5. Greedy backward selection evaluates multiple candidate masks simultaneously at each removal step, introducing a winner's-curse bias: the selected mask's greedy score is an upward-biased estimate of its true expected return. To correct for this, every mask on the final backward-selection path is re-evaluated with an independent episode sample after selection is complete. For Baoding and Bounce approximately 3 400 independent episodes are used per K level; for Multi-obj. Rotate and Forest Run approximately 500 episodes are used. All K-vs-return curves reported in the paper use these re-evaluated means.

D Environment Specification Details

D.1 Multi-object Rotate

The Multi-obj. Rotate task uses a 16-joint Allegro hand to rotate an object about the vertical axis without dropping it. The object set contains 16 shapes, including blocks, cylinders, and irregular objects, visualized in fig. 11. Sensor selection operates over 16 tactile locations spread across the fingers and palm. This task is based on Yin et al. (2023), but ported to Isaac Lab and with the arm removed.



Figure 11: **Multi-obj. Rotate:** 16 object geometries.

Observations. Table 7 lists dimensions per observation key. The policy uses four frames of history. Proprioception and the rotation command remain available at every sensing budget; only tactile readings are selectable. Object state is provided as privileged information to the critic and auxiliary reconstruction, but not the actor.

Table 7: Multi-object Rotate observations per frame.

Term	Dim.	Availability	Description
Joint positions	16	Always available	Normalized hand joint angles.
Previous targets	16	Always available	Normalized hand joint position targets.
Rotation axis	3	Always available	Three-dimensional target rotation axis (fixed).
Tactile forces	16	Selectable	Contact-force magnitude at each sensor.
Object state	13	Privileged	Position (3), quaternion (4), linear velocity (3), and angular velocity (3).

Actions. The policy outputs 16 relative joint commands. Commands are exponentially smoothed with coefficient 0.8, scaled by 0.2, and added to the previous position targets. The scale receives a per-step perturbation of $\pm 5\%$. Targets are clipped to joint limits and tracked by a PD controller.

Reward. The reward is the weighted sum in Table 8. Here $\Delta\theta$ is the signed rotation increment about the commanded axis, d_i is the distance from fingertip i to the object center, and $\boldsymbol{\tau}$ and $\dot{\mathbf{q}}$ refer to the 16 hand joints.

Training conditions. Isaac Lab simulates physics at 120 Hz and applies policy actions at 10 Hz. Episodes last at most 500 control steps (50 s), ending earlier on a drop, excessive tilt (0.4π), or object displacement past the local $x = 0.59$ m boundary. Training randomizes object mass (0.2–0.6 kg), friction (0.2–3.0), scale (0.95–1.10), controller parameters, and initial position, and applies observation/action noise and external force perturbations.

Table 8: Multi-object Rotate reward.

Term	Definition/Equation	Weight
Rotation	$\mathrm{clip}(\Delta\theta, -\pi, \pi)$	20
Fingertip proximity	$\frac{1}{4}\sum_{i=1}^{4} \mathrm{clip}\left(\frac{0.1}{4d_i+0.02}, 0, 1\right)$	0.1
Object translation	$\|\mathbf{v}_{\mathrm{obj}}\|_2$	-0.1
Torque	$\|\boldsymbol{\tau}\|_2^2$	-0.0003
Mechanical work	$\sum_{j=1}^{16} \|\tau_j \dot{q}_j\|$	-0.0003
Drop	$\mathbf{1}[\|\mathbf{p}_{\mathrm{obj}} - \mathbf{p}_{\mathrm{goal}}\|_2 \geq 0.10]$	-50

D.2 Baoding and Bounce

Baoding and Bounce use a 20-actuator Shadow hand with 20 selectable joint inputs and 17 selectable tactile sensors. Baoding repeatedly exchanges the positions of two balls in the palm; each ball has mass 55 g and diameter 3.81 cm. Bounce repeatedly tosses and catches a single ball of mass 30 g and diameter 7 cm. These tasks are part of the Robot Tactile Olympiad (Miller et al., 2025).

Observations. Both tasks use the observations in Table 9 with four frames of history. For each of the 20 joints, the position, velocity, position error, and previous action are selected together. The 17 tactile readings are selected individually. Ball state is used only for auxiliary reconstruction.

Table 9: Baoding and Bounce observations per frame.

Term	Dim.	Availability	Description
Joint positions	20	Selectable	Normalized actuated joint angles.
Joint velocities	20	Selectable	Normalized actuated joint velocities.
Position errors	20	Selectable	Commanded minus measured joint positions.
Previous actions	20	Selectable	Coupled to the corresponding joint's mask.
Tactile contacts	17	Selectable	Binary contact indicators (force magnitude > 0.01 N).
Ball state	13 / 7	Reconstruction only	Baoding: two positions (6), two velocities (6), and separation (1). Bounce: position (3), velocity (3), and speed (1).

Actions. Both policies output 20 normalized joint position commands.

Baoding reward. Each ball earns reward for approaching its target. Once both balls have come within 1 cm of their targets, the policy receives a completion reward and the targets swap. Table 10 defines the terms: d_i is ball i's distance to its target, and $g = 1$ when both have reached their targets since the previous swap.

Table 10: Baoding reward.

Term	Definition/Equation	Weight
Target proximity	$\sum_{i=1}^{2}[1 - \tanh(d_i/0.1)]$	0.1
Target completion	g	10

Bounce reward. The reward is 10 when the ball recontacts the hand after leaving it and remaining out of contact for at least five control steps, and zero otherwise.

Training conditions. Both tasks use Isaac Lab with physics at 240 Hz and control at 60 Hz. Episodes last at most 600 control steps (10 s). Resets perturb joint positions by up to 0.125 rad.

Baoding episodes end early if either ball falls below 0.3 m or the balls separate by at least 0.15 m. Ball positions are perturbed by up to 5 mm per axis at reset. Bounce episodes end early if the ball falls below 0.4 m or moves more than 0.2 m laterally from its reference y coordinate. Its ball position is perturbed by up to 1 cm per axis at reset.

D.3 FOREST RUN

Forest Run uses a Unitree Go1 quadruped to move forward through a randomly sampled tree field while maintaining a stable gait. The field contains 256 trees, randomized per-episode, over a 32×32 m region. To speed up simulation, the trees are modeled in code rather than as physics objects. Sensor selection is performed over 20 head-mounted range rays. Figure **??** shows the environment and ray arrangement.

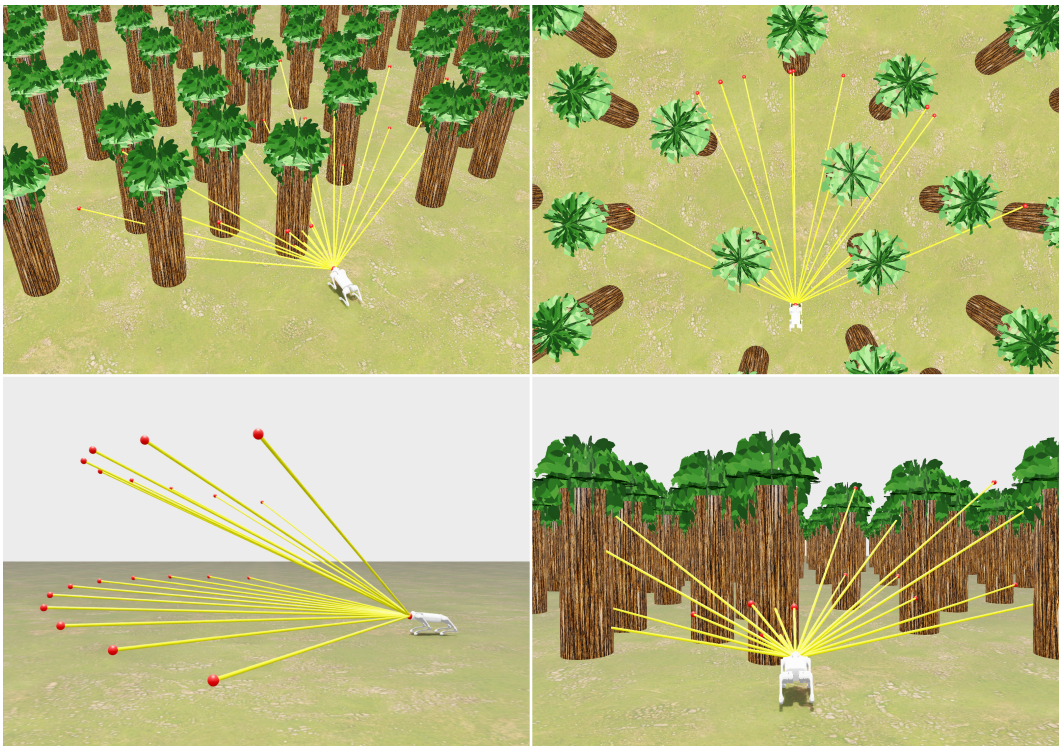


Figure 12: Forest Run environment, shown from oblique (upper left), top (upper right), side (lower left), and rear (lower right) views. 20 LiDAR rays form two sensor arcs at 0° and 20° elevation relative to the robot's body.

Observations. Table 11 lists observations per frame. The 46-dimensional proprioceptive input is always available. Ray distances are clipped at 5 m.

Table 11: Forest Run observations per frame.

Term	Dim.	Availability	Description
Joint positions	12	Always available	Normalized joint angles.
Joint velocities	12	Always available	Normalized joint velocities.
Base velocities	6	Always available	Linear and angular velocity in the body frame.
Projected gravity	3	Always available	Unit gravity vector in the body frame.
Previous actions	12	Always available	Previous joint commands.
Heading	1	Always available	Base yaw relative to the fixed forward direction.
Ray distances	20	Selectable	11 horizontal rays and 9 upward-pitched rays (20°).
Unmasked rays	20	Privileged	Never masked, for reconstruction target.
Foot contacts	4	Privileged	Binary contact indicators (force > 1 N).
Foot air times	4	Privileged	Time since each foot last contacted the ground.

Actions. The policy outputs 12 joint commands.

Reward. The first six terms in Table 12 are multiplied, a design inspired by Ma et al. (2024). The obstacle penalties are then added. Let h be base height, $\mathbf{g}_b$ projected gravity, and d the minimum horizontal distance between a foot and an obstacle center. A 2.5 Hz clock sets the desired foot-contact pattern: the front-right and rear-left feet move together, alternating with the other two feet. Let c_i indicate whether foot i touches the ground, and let $\hat{c}_i$ be its desired contact signal. We compute $\hat{c}_i = \sigma(\cos(2\pi\phi_i)/0.1)$ using phase offsets $(0, 0.5, 0.5, 0)$ for the front-right, front-left, rear-right, and rear-left feet, respectively.

Table 12: Forest Run reward. The upper block contains multiplicative factors.

Term	Definition/Equation	Weight
Forward speed	$\exp(-\lvert v_x - 2\rvert)$	1
Base height	$\exp(-5\lvert h - 0.34\rvert)$	1
Upright posture	$\exp(-5\lVert \mathbf{g}_b - (0,0,-1)\rVert_2)$	1
Joint limits	$\mathbf{1}$[all joints within limits]	1
Action smoothness	$\exp(-0.1\lVert \mathbf{a}_t - \mathbf{a}_{t-1}\rVert_2)$	1
Trot tracking	$\exp(-2 \cdot \frac{1}{4}\sum_i \lvert \hat{c}_i - c_i\rvert)$	1
Obstacle proximity	$[\mathrm{clip}((0.75 - d)/0.5, 0, 1)]^2$	-1
Lateral boundary	$\mathbf{1}[\lvert y - y_0\rvert > 16]$	-1
Near obstacle	$\mathbf{1}[d \le 0.30]$	-1

Training conditions. Isaac Lab runs physics at 200 Hz and control at 50 Hz. Episodes last at most 1,000 control steps (20 s) and terminate if the robot flips over, its trunk touches the ground, its base falls below 0.15 m, it crosses the lateral boundary, or it comes within 0.30 m of an obstacle. At reset, we randomize the obstacle layout, gait phase, and joint positions. Specialists train for 40M transitions without obstacles, followed by 160M with obstacles.

E Additional Experimental Results

E.1 Discovered Sensor Specifications

Across tasks, our method identifies non-intuitive, often compact, task-relevant sensor subsets, and we find the configurations often contradictory to human intuition and common heuristics used to measure a sensor's importance. Selected specifications are visualized in fig. 6; see figs. 13 to 16 for comprehensive visualizations.

For **Multi-obj. Rotate**, the thumb-tip sensor is retained until the final elimination step, consistent with the thumb's primary role in rotating the object. Sensors on the distal thumb, proximal ring finger, middle fingertip, and palm center were preserved next. Even after all 16 tactile sensors were removed, the policy retained 88% of performance (mean return decreased from 1806 to 1597), suggesting that touch sensing only moderately contributes to policy performance. Activation frequency was not significantly associated with removal order (Spearman $\rho = 0.35$, $p = 0.19$; Figure 18); thus, frequently activated sensors are not necessarily the most informative for the policy.

Baoding and **Bounce** prioritize different sensing modalities: Baoding favors proprioception, whereas Bounce retains tactile sensing until much sparser budgets. Baoding removes all tactile sensors by $k = 9$ while retaining 90% of full-sensor performance. It instead preserves proprioception from the thumb base, wrist, and finger-base joints needed for the coordinated motion of circulating the balls. Performance declines sharply as the final joints are removed. In contrast, Bounce retains tactile sensors on the thumb and ring finger later than most joints, using these contacts to trap the ball within its grasp. Nevertheless, a small proprioceptive core remains essential for bouncing. Three joints—including the wrist, which drives the repetitive bounce motion—retain 97% of full-sensor performance; removing the final wrist observation at $k = 0$ reduces performance below 1%.

For **Forest Run**, eight rays retain 90% of full-sensor performance. Backward selection first favors the upper-center ray, which detects obstacles directly ahead, followed by rays oriented toward the upper right. This asymmetry aligns with the robot's mean rightward displacement of $10.02\,\mathrm{m}$ across $4{,}096$ rollouts (SD $3.84\,\mathrm{m}$; Figure 17), placing these rays along its typical path through the trees.

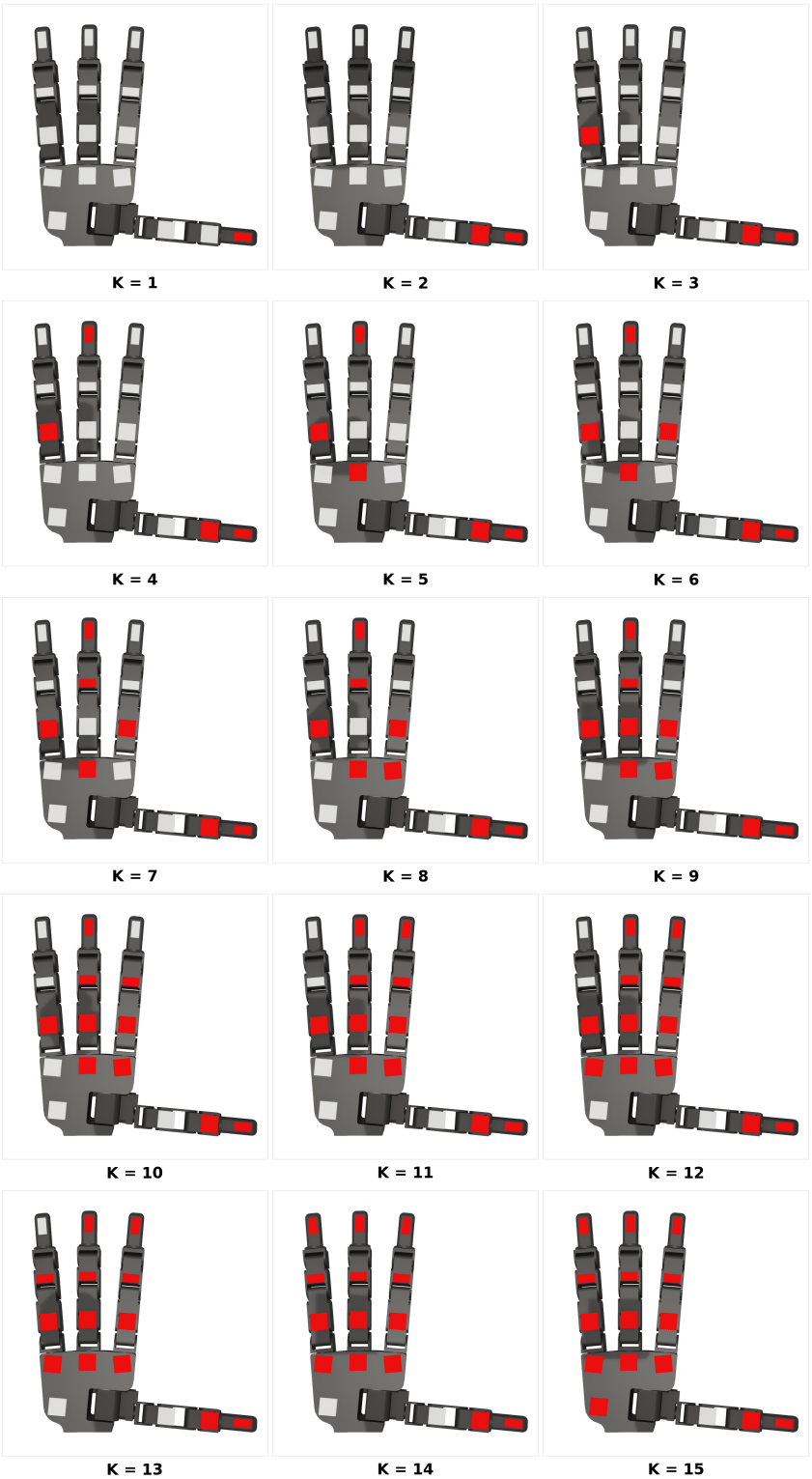


Figure 13: **Multi-obj. Rotate specifications discovered by backward selection.** Each panel shows the selected tactile sensors in red at budget K; unselected sensors are white.

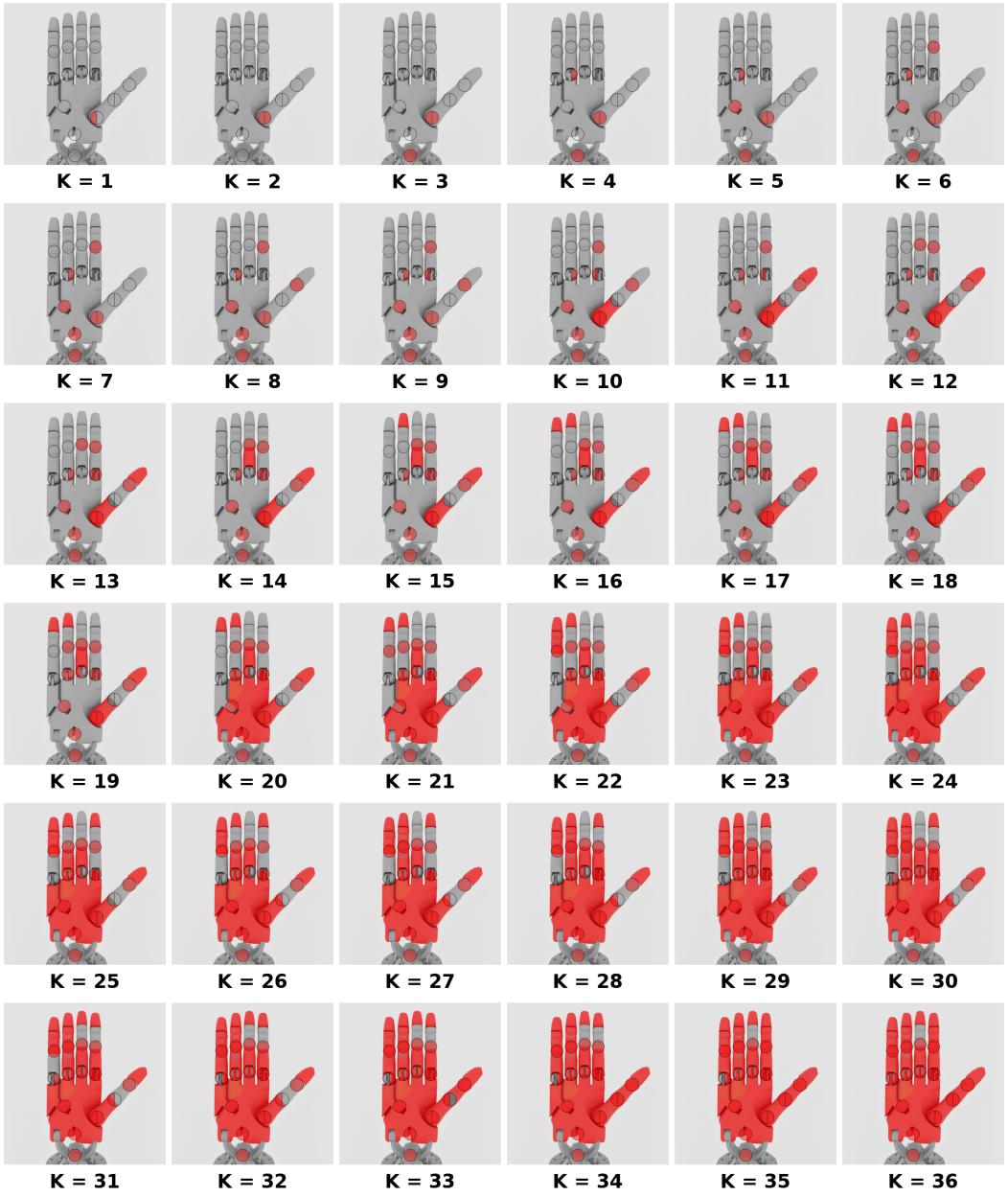


Figure 14: **Baoding specifications discovered by backward selection.** Selected inputs are red and unselected inputs are gray. Joint observations are represented as circles; split circles show rotational (left) and flexion (right) degrees of freedom. Selected touch sensors are red bodies.

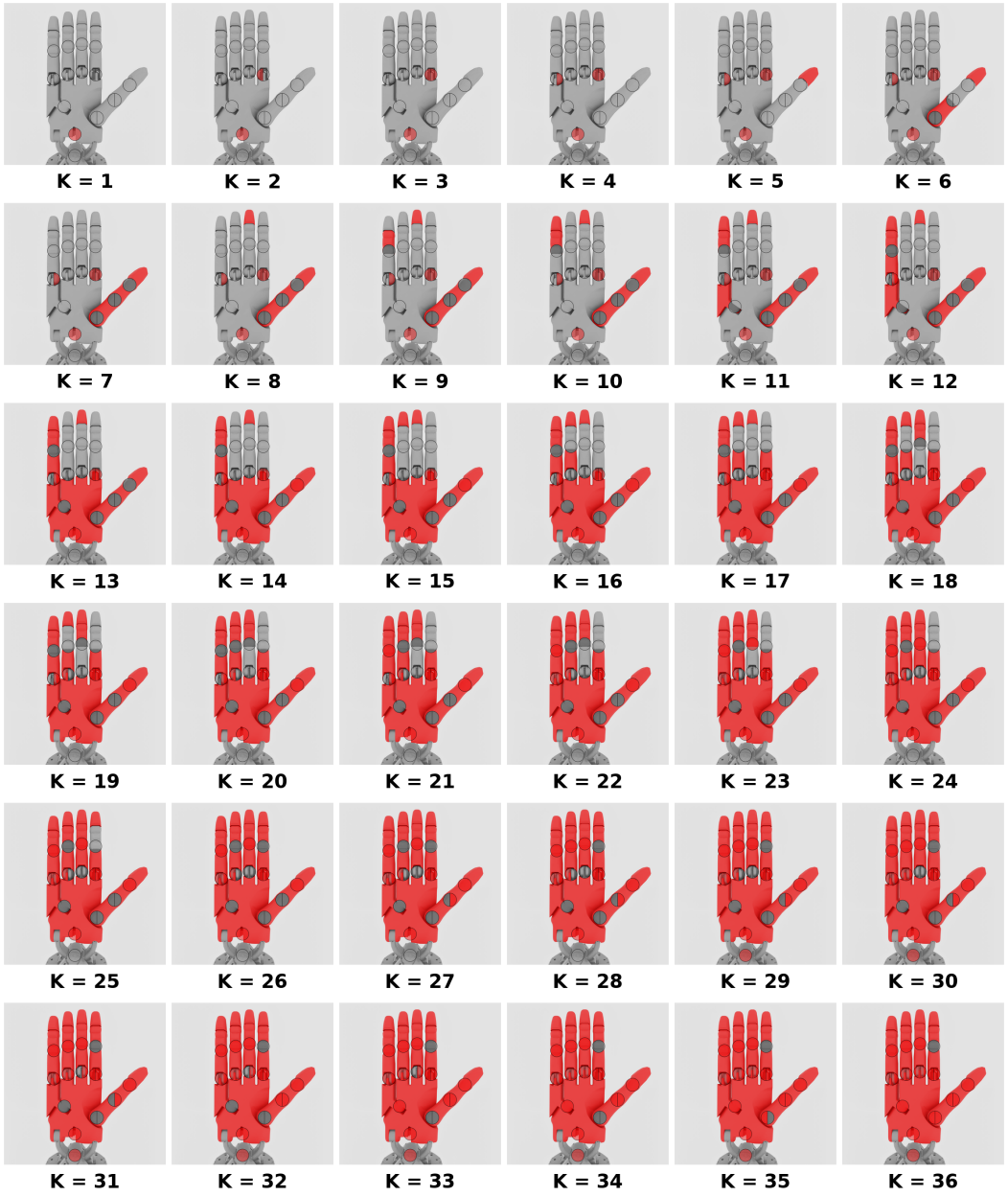


Figure 15: **Bounce specifications discovered by backward selection.** Selected inputs are red and unselected inputs are gray. Joint observations are represented as circles; split circles show rotational (left) and flexion (right) degrees of freedom. Selected touch sensors are red bodies.

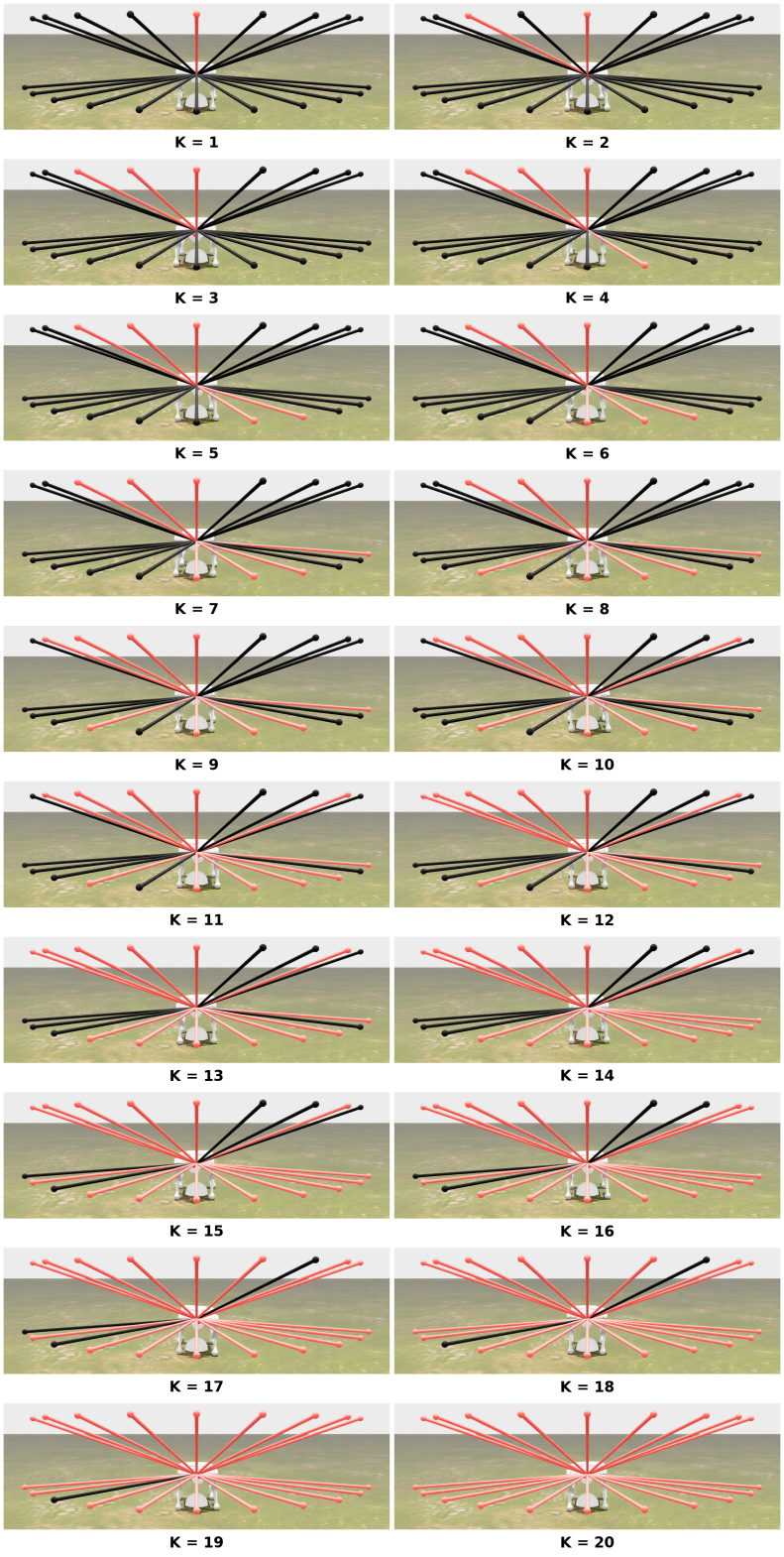


Figure 16: **Forest Run specifications discovered by backward selection.** Each panel shows the selected range-sensor rays in red at budget K; unselected rays are black.

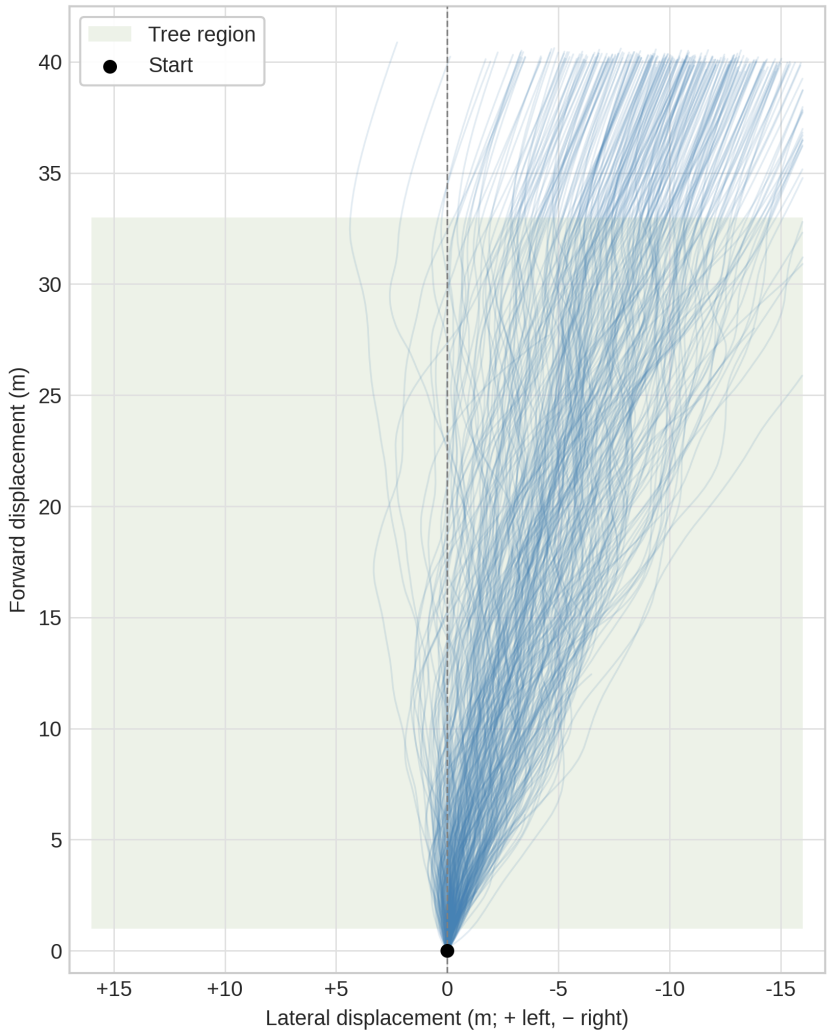


Figure 17: **Forest Run rollout trajectories.** Across 4,096 rollouts, the robot exhibits a mean rightward displacement of $10.02\,\mathrm{m}$ (SD $3.84\,\mathrm{m}$).

E.2 Sensor Activation Frequency

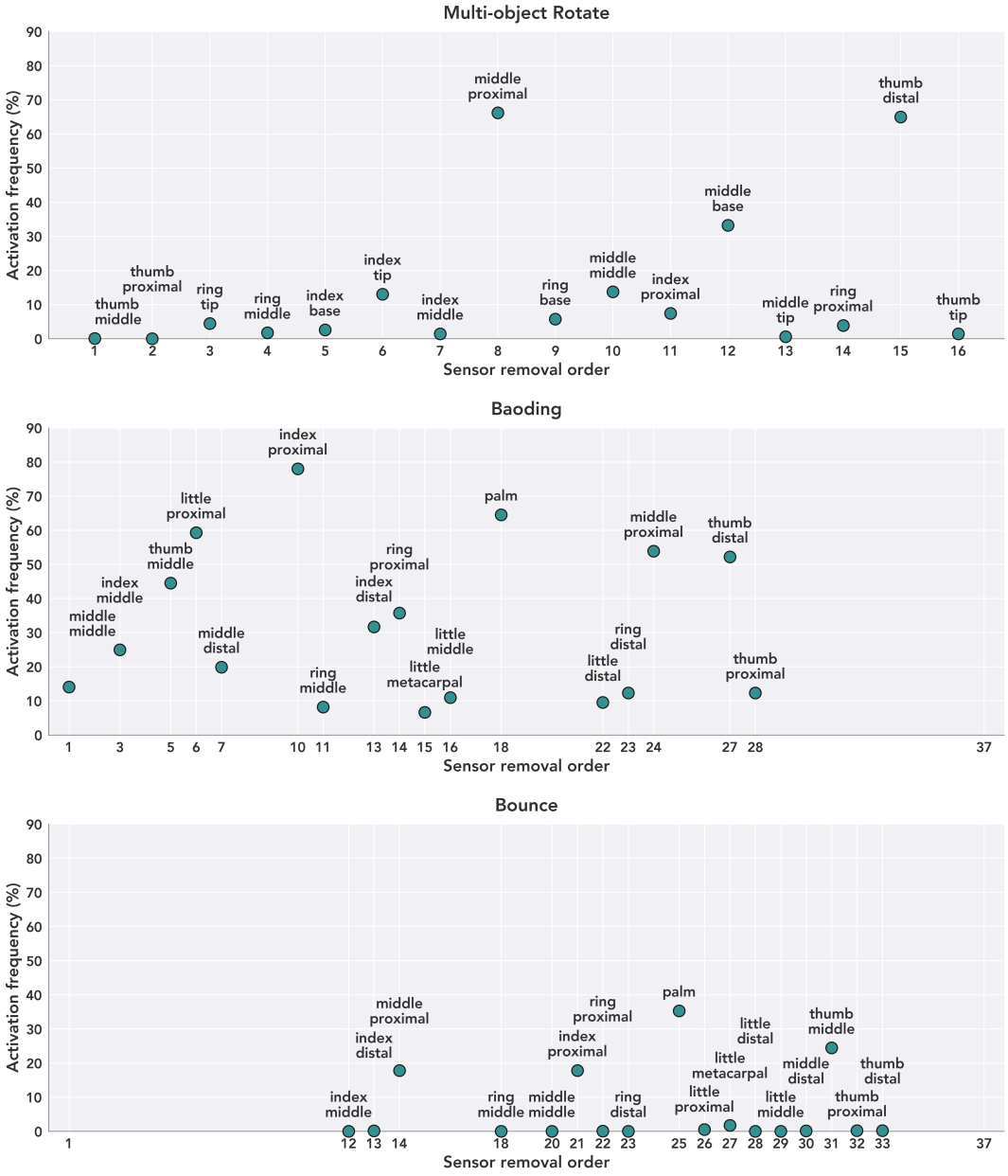


Figure 18: Touch-sensor activation frequency versus backward-selection removal order for Multi-object Rotate, Baoding, and Bounce. Higher removal orders indicate sensors retained longer; gaps in Baoding and Bounce correspond to joint removals. Frequently activated sensors are not consistently retained longer.

F Backward Selection vs. Evolutionary Search

We consider evolutionary search as an alternative to the backward selection procedure for finding high performing sensor specifications. Evolutionary search works even if the optimization problem is discrete, combinatorial, and non-differentiable. Its use is further enabled by our GPU-accelerated simulator, which makes rollout-based evaluation of many candidates practical in parallel.

Search procedure. We maintain a population of N sensor specifications, each satisfying the sensing level constraint $\|\sigma\|_0 = k$. At each search iteration, every candidate is evaluated by running the generalist $\pi_{\hat{\theta}}(a \mid o^\sigma)$ for n episodes and estimating its fitness as the average return. The top E candidates are retained as elites. A new population is then generated from these elites using crossover and mutation, while preserving the fixed sensor budget.

For crossover, we sample two elites and construct a child mask by selecting k active sensors from their union. For mutation, each active sensor is independently replaced with a randomly chosen inactive sensor with probability μ. Repeating this process over multiple iterations gradually shifts the population toward masks with high value conditioned on the generalist. The evolutionary search is summarized in fig. 19.

Algorithm 1: Evolutionary Search over sensor specifications at level k

Input: Number of candidate sensors d, sensor budget k, policy π_θ,
population size N, elite size E, mutation rate μ, crossover rate γ,
episodes per candidate n, search iterations T.

```
Initialize population C_0 = {σ_0^(i)}_{i=1}^N with σ_0^(i) ∈ Σ_k
For t = 1, ..., T:
  Evaluate: For each σ ∈ C_{t-1}, estimate V(σ) by averaging
    the return of π_θ(· | o^σ) over n episodes
  Select elites: Let E_t be the top E candidates in C_{t-1} by Ṽ(σ)
  Generate next population: Initialize C_t ← ∅
  While |C_t| < N:
    Sample parent σ ∈ E_t
    With probability γ, sample parent σ' ∈ E_t, recombine σ, σ' into child c ∈ Σ_k
    Else set c ← σ
    Mutate c with rate μ while preserving ||c||_0 = k
    Add c to C_t
  End While
End For
```

Output: Top-performing masks from final population $\mathcal{C}_T$.

Figure 19: Evolutionary search procedure for generating high scoring sensor specifications at sensing level k.

Because evolutionary search can propose distinct candidates, compared to backward selection whose candidates all derive from the same parent, we expect evolutionary search to outperform backward selection, at the cost of additional compute.

To verify this, we compared evolutionary search against backward selection in the Baoding task in the low k regime, using an earlier checkpoint of our generalist policy. In fig. 20, we can see that evolutionary search does find higher scoring specifications than backward selection, especially in the lower sensing ranges. However, we found that backward selection was simpler, faster, and less resource demanding to run, with little degradation at higher sensing levels, so we use backward selection by default.

G Real World Validation

We train specialist policies in simulation and deploy them on a real LEAP hand (Shaw et al., 2023) on different specifications using the mature sim-to-real policy training pipeline in MuJoCo Playground (Zakka et al., 2025). To keep comparisons fair, the policy training pipeline is fixed across policies (same hyperparameters, domain randomization parameters, and training budget), allowing us to isolate the performance difference between policies to their different touch configurations.

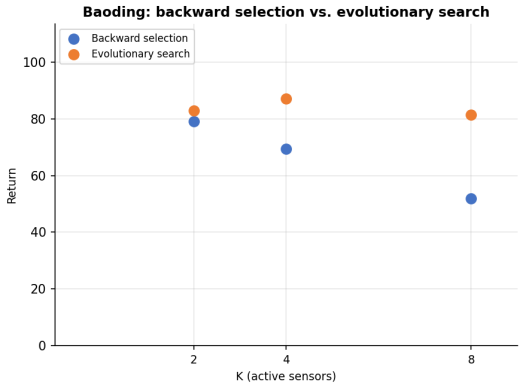


Figure 20: Evolutionary search outperforms backward selection in low sensing regimes, but requires more compute.

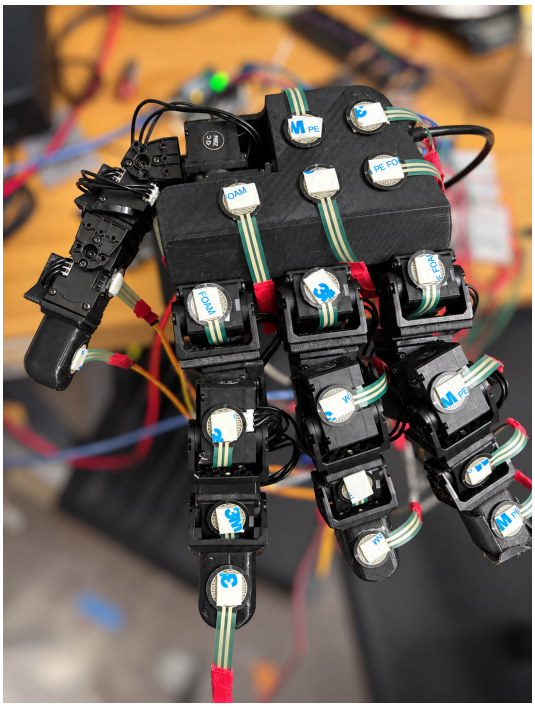


Figure 21: The LEAP hand instrumented with 20 FSR sensors.

G.1 Real robot setup

We use the LEAP hand (Shaw et al., 2023), a 16-degree-of-freedom anthropomorphic hand with four fingers (index, middle, ring, thumb), each actuated by four Dynamixel servos. The manipulated object is a cube of size 7 cm.

Motor control. The Dynamixel servos are driven over a single 4 Mbaud bus in position–current control mode, which tracks a commanded position while capping the motor current at 350 mA to prevent overheating and excessive grip force. We use the proportional–integral–derivative gains from the LEAP hand sim-to-real pipeline of MuJoCo Playground (Zakka et al., 2025), obtained through correspondence with the authors: $k_P = 300$, $k_I = 0$, $k_D = 100$ (Dynamixel register units), applied uniformly to all 16 joints.

Control loop. The policy runs at 20 Hz (control interval 50 ms), matching the simulation control timestep. Each control step performs one bulk read of present position, velocity, and current for all 16 servos, reads the latest tactile frame, builds the observation, queries the policy, and writes the new joint targets. Joint targets are computed exactly as in simulation, $q^{\text{target}} = q^h + 0.6a$, where a is the policy action and q^h is the home pose used during training; targets are clipped to the joint limits. The observation consists of the 16 joint positions, the previous action, and the K active binary touch channels.

Energy measurement. To compare the effort expended by different policies, we estimate mechanical work from the servos' built-in telemetry. At every control step t we record the present current $I_{i,t}$ (in mA, via the Dynamixel present-current register, 1.34 mA per unit) and present angular velocity $\omega_{i,t}$ (rad/s) for each joint i, together with the measured step duration Δt_t. We convert current to torque using a nominal torque constant $k_\tau = 0.55\,\text{N·m/A}$ and integrate the resulting power over the episode:

$$W = \sum_t \sum_{i=1}^{16} k_\tau \, I_{i,t} \, \omega_{i,t} \, \Delta t_t. \tag{16}$$

Instrumenting the touch sensors. We adapt the tactile sensing setup from (Yin et al., 2023), where 20 Force Sensing Resistors (FSR) and their continuous force readings are measured by a program on an Arduino board, see fig. 22. The hand is instrumented with 20 force-sensitive resistors (FSRs): 8 on the palm and 12 on the fingers (proximal, middle, and tip pads of the index, middle, and ring fingers, and proximal, distal, and tip pads of the thumb).

A multiplexer is used to handle the routing between the sensors and the Arduino. We found that cross-talk noise was an issue - sensors with neighboring pins on the multiplexer could affect each other's readings. This was alleviated through adjusting a threshold. On the host, each channel is binarized with a fixed threshold, matching the binary touch observation used in simulation. Per-channel thresholds were set by hand after observing the no-contact baseline of each pad.

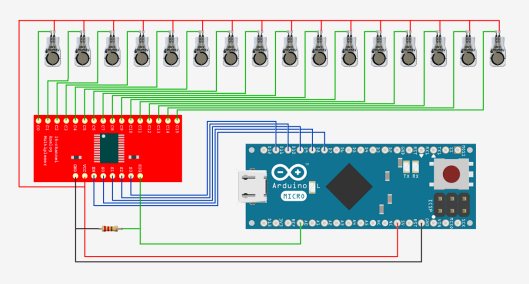


Figure 22: The circuit design for the tactile sensors, to be mounted on the hand.

Training and Evaluation. To acquire our set of candidate masks for real-world evaluation, we largely mirror the experimental procedure described earlier to train the generalist policy and acquire the final set of high performing masks.

A generalist policy is first trained in simulation to rotate objects from a set of 7 cube variants. Then evolutionary search is performed to find high scoring candidate sets, using the following hyperparameters $N = 256$, top $E = 128$ elites, $T = 50$ iterations, mutation rate $\mu = 0.1$, crossover $\gamma = 0.1$. We choose to find sensor specifications with $K = 8$ sensors, and they are visualized in fig. 23. The evolved specifications place sensors on the thumb, outer areas of the palm, and on the parts of the finger closest to the palm. We believe this placement helps the policy localize the 7cm cube.

Then, we run MuJoCo Playground's sim-to-real pipeline for the LEAP hand rotation task, which uses PPO to train a policy to rotate a cube using a reward function with angular velocity and termination terms, see table 13.

Baselines. We evaluate the sim-to-real performance of policies trained on the following sensor configurations:

Table 13: PPO hyperparameters for the LEAP cube-rotation specialist policies deployed on the real robot.

Parameter	Value
Optimization	
Environment steps	3×10^8
Parallel environments	8192
Unroll length	40
Minibatches per batch	32
Updates per batch	4
Minibatch size	256
Learning rate	3×10^{-4}
Discount factor γ	0.97
GAE λ	0.95
PPO clipping ϵ	0.3
Entropy coefficient	0.01
Reward scaling	1.0
Observation normalization	Yes
Seeds per configuration	3
Networks	
Policy MLP	(512, 256, 128), SiLU
Value MLP	(512, 256, 128), SiLU
Policy input	joint pos. (16) + prev. action (16) + K touch bits
Value input	privileged state (113-d; adds joint vel., torques, fingertip pos., cube pose and velocities)
Action distribution	Tanh-squashed Gaussian, learned scalar std
Environment	
Control frequency	20 Hz ($\Delta t_{\text{ctrl}} = 0.05$ s)
Physics timestep	0.01 s (5 substeps)
Episode length	500 steps (25 s)
Action scale	0.6
Joint-position observation noise	$\mathcal{U}(-0.05, 0.05)$ rad
Reward	$1.0 \cdot \omega_z^{\text{cube}}$, -100 on cube drop
Domain randomization (uniform ranges)	
Fingertip sliding friction	$[1.5, 2.5]$
Actuator k_p scale	$[1.2, 2.0]$
Joint damping scale	$[0.3, 0.8]$
Joint friction-loss scale	$[0.2, 0.8]$
Cube inertia scale	$[0.8, 1.2]$
Cube CoM offset	$[-5, 5]$ mm per axis
Hand mass scale	$[0.9, 1.1]$
Joint armature scale	$[1.0, 1.05]$
Initial joint offset	$[-0.05, 0.05]$ rad

- **Ours.** The top 3 touch sensor configurations from our evolutionary search with $k = 8$ active sensors.
- **Random.** 3 random configurations with $k = 8$ activated sensors.
- **Fingers.** We conducted a literature search on tactile sensor placement for dexterous hands and found a recurring high-performing configuration where tactile sensors are concentrated on fingers and sparse on the palm (Lee et al., 2024; Guo et al., 2024; Tang et al., 2025). Note this configuration uses 13 sensors, which is more than ours or random.
- **Proprio.** The policy does not have any touch sensors enabled, relying only on proprioception.

Results. We evaluate the policies with two metrics. First, we count the number of cube rotations (90° turns) performed in an episode of length 500 steps (roughly 25 seconds), and the energy usage. 15 trials are performed for each configuration, and we report the mean and standard error over the trials in fig. 8.

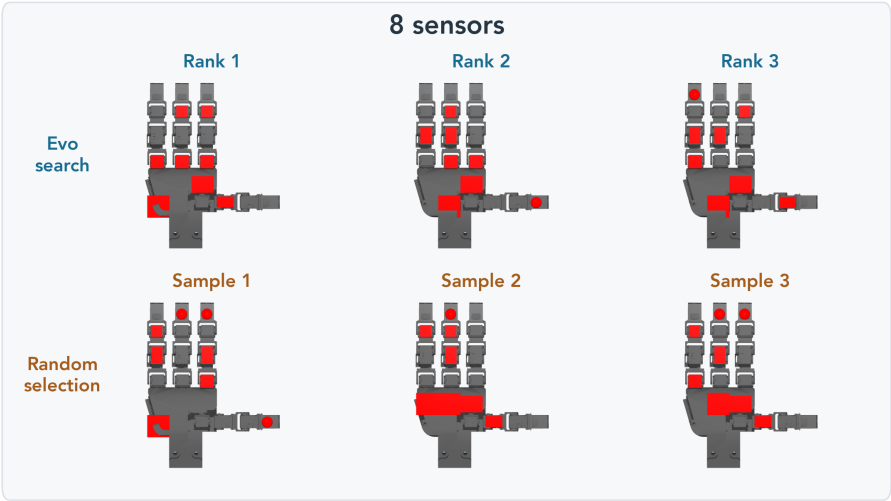


Figure 23: **Real-world sensor specifications.** Top-ranked evolutionary-search and sampled random configurations on budget $k = 8$. Selected tactile sensors are shown in red.

As seen in figure 8, the 3 evolved configurations have higher rotation than all baselines, and two out of three evolved configurations have the lowest energy of all configurations. We observed interesting qualitative differences between the policies. The evolved configuration policies exhibited a more deterministic, reactive behavior towards spinning the cube. In contrast, we found that random and proprioception policies display repetitive, open-loop finger motions with large magnitudes. These open-loop behaviors would often result in the cube slipping or getting stuck, reducing the amount of total rotations in the episode. The baselines would also move the fingers regardless of the particular pose of the object - resulting in wasteful movements and increased energy usage.

We also found the Fingers configuration to underperform. It was reported to do well for dexterous hands (Allegro, Shadow) trained in dexterous rotation tasks in Isaac and Gymnasium environments in prior works Lee et al. (2024); Guo et al. (2024). However, our setup still differs in major factors like morphology (we use a LEAP hand) which could result in different contact dynamics.