Poke and Strike: Learning Task-Informed Exploration Policies

Marina Y. Aoyama[1], **João Moura**[1], **Juan Del Aguila Ferrandis**[1], **Sethu Vijayakumar**[1]
[1]School of Informatics, The University of Edinburgh, UK

Abstract: In many dynamic robotic tasks, such as striking pucks into a goal outside the reachable workspace, the robot must first identify the relevant physical properties of the object for successful task execution, as it is unable to recover from failure or retry without human intervention. To address this challenge, we propose a task-informed exploration approach, based on reinforcement learning, that trains an exploration policy using rewards automatically generated from the sensitivity of a privileged task policy to errors in estimated properties. We also introduce an uncertainty-based mechanism to determine when to transition from exploration to task execution, ensuring sufficient property estimation accuracy with minimal exploration time. Our method achieves a 90% success rate on the striking task with an average exploration time under 1.2 seconds—significantly outperforming baselines that achieve at most 40% success or require inefficient querying and retraining in a simulator at test time. Additionally, we demonstrate that our task-informed exploration rewards capture the relative importance of physical properties in two manipulation tasks and the classical CartPole example. Finally, we validate our approach by demonstrating its ability to identify object properties and adjust task execution in a physical setup using the KUKA iiwa robot arm. The project website is available at marina-aoyama.github.io/poke-and-strike/.

Keywords: Interactive Perception, Manipulation, Reinforcement Learning, System Identification

1 Introduction

Exploratory motions are essential for identifying system parameters and adjusting actions accordingly when executing interactive tasks. Unlike visual properties, identifying physical properties—such as friction, mass, weight distribution, and restitution—requires active interaction with objects. For example, a robot might push an object to identify friction for striking [1], poke a sponge to evaluate stiffness for wiping [2], or shake [3] or stir [4] liquids to determine viscosity for pouring. While such exploratory behaviors naturally emerge in humans [5], enabling robots to autonomously explore through physical interactions remains a significant challenge.

Task-informed exploration. A key challenge in achieving exploratory behaviors is coming up with informative exploration strategies that identify task-relevant properties. Most existing approaches rely on pre-defined, human-designed exploratory motions tailored to specific tasks [3, 4, 6, 2, 7, 8], which can be suboptimal and cumbersome to obtain for every new task. Alternatively, Memmel et al. [1] propose learning informative exploratory motions given a set of properties of interest, while Liang et al. [9] introduce exploration policy learning guided by task information using reinforcement learning (RL), implicitly prioritizing the estimation of task-relevant properties. However, both approaches [1, 9] require inefficient simulator queries and re-optimization of task motions for each object with the identified properties at test time. Our approach, in contrast, enables immediate task execution after exploration, in addition to learning exploratory motions to identify task-relevant properties using rewards automatically generated from the task policy.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

Uncertainty-based policy switching. Another challenge lies in the transition from exploration to task execution. Prior approaches often rely on a fixed-length exploration phase [1, 9], which may be either too short, leaving the robot with insufficient information, or excessively long, which is inefficient. To autonomously switch from exploration to task execution, the robot needs sufficiently accurate property estimates for a given task. However, the estimation accuracy is inaccessible outside of the simulated environment. Additionally, the robot must determine how accurately it needs to estimate each property to ensure task success. To address this, our approach computes uncertainty estimates and, in simulation, we determine the uncertainty thresholds based on task completion.

In summary, the contributions of this work are:

- We propose a task-informed exploration RL approach that trains an exploration policy using rewards automatically generated from the sensitivity of a privileged task policy to errors in estimated properties, leading to the identification of task-relevant properties.
- We demonstrate that autonomously discovered exploration rewards lead to improved task performance by more accurately identifying relevant properties.
- We introduce the simultaneous learning of an online estimator with the exploration policy, enabling estimation within distribution of the exploratory observed states.
- We present a method that autonomously transitions from exploration to task execution using uncertainty in property estimates, with thresholds computed from task success.
- We validate our approach on a physical robotic setup, showing that it successfully adjusts task motions for different object properties.

2 Problem Statement

We address the problem of performing **one-shot robotic tasks** that involve interacting with objects or environments with **unknown physical properties**, where the robot is **unable to recover from failure or retry** without human intervention. Examples include striking pucks with varying friction to an unreachable goal, tossing objects with different centers of mass, or shooting a basketball with unknown restitution. These tasks require **identifying the physical properties of the manipulated object prior to task execution**, as initiating the task with incorrect assumptions about the properties can lead to irreversible failure—making online adaptation or correction unsuitable.

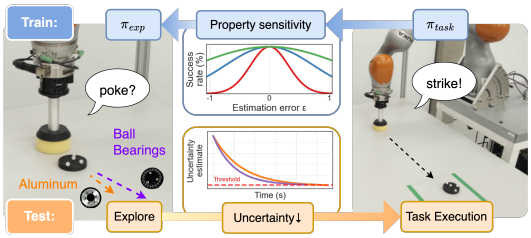


Figure 1: **Task-informed exploration** approach enables the robot to autonomously learn how to explore and identify task-relevant properties by modeling task sensitivity to each property. For dynamic tasks such as striking, the robot must first identify the object's properties through exploratory motions to achieve the task success, avoiding irreversible failure.

In this work, we seek to develop a method that addresses the following questions: (1) how to automatically discover informative exploratory motions for identifying task-relevant properties, when these motions differ from optimal task motion and vary depending on the task; (2) how to adaptively determine the timing to transition from exploration to task execution, ensuring sufficient property estimation accuracy for a given task while minimizing exploration time; and (3) how to execute the task immediately after exploration, without additional simulation queries or retraining at test time.

3 Related Work

Domain randomization. Domain randomization [10, 11] is a powerful technique in RL for enhancing the robustness of machine learning models and policies by introducing variability in environmental factors, such as the visual and physical properties of objects. While this approach improves

the robustness of learnt policies across diverse environmental conditions, it often results in policies that exhibit average behavior [12, 1]. To adapt behavior based on the physical properties of objects, Ramos et al. [12] propose non-uniform domain randomization. However, this method requires obtaining the posterior distribution of object properties and retraining the task policy for those specific properties each time the robot executes the task. On the other hand, Peng et al. [13] use Recurrent Deterministic Policy Gradient (RDPG) in RL, aiming to infer dynamics by capturing temporal dependencies. While inspired by these methods, we find that they alone are insufficient for tasks requiring exploratory motions to identify physical properties prior to task execution.

System Identification. System identification [14] is crucial for modeling system dynamics from observed data. A key challenge is generating exploratory motions that yield informative observations for parameter identification. Many prior approaches rely on predefined motions [15, 7, 6, 8, 3, 4, 16]. Others iteratively update parameters through repeated task executions [17, 18]. However, optimal task motions may differ from those most informative for parameter identification. To address this, prior works optimize exploratory motions by minimizing property estimation errors [19], reducing state transition prediction errors [20], or maximizing Fisher information to generate observations that most effectively distinguish differences in properties [1]. However, these methods assume equal importance of all given parameters and fail to identify task-relevant ones.

Liang et al. [9] introduce task-oriented exploration using RL, where task rewards implicitly guide the exploration policy to focus on properties critical for task success. However, their approach incurs inefficient simulation queries and requires a fixed exploration duration to infer parameters by matching simulated and real-world observations, aiming to mitigate the out-of-distribution issue with learned estimators [19]. In contrast, our method trains the online property estimator simultaneously with the exploration policy to align the training data distribution, while enabling adaptive exploration length based on uncertainty estimates. Additionally, these prior works [9, 1] require re-optimization of the task policy at test time. Our method enables immediate task execution after exploration using an online property estimator and a privileged task policy. Dass et al. [21] also propose a dual-policy framework for task and information-seeking, but assume minimal action space overlap—precluding applicability in scenarios where exploratory and task actions interfere, such as physical property identification. In contrast, as summarized in Table 1, our method enables one-shot task execution immediately after exploration, with automatically determined exploration time, and without constraints on action spaces.

Privileged learning for robot control. Privileged learning [22] leverages information inaccessible during real-world deployment. This privileged information accelerates the learning process by providing the agent with critical knowledge, such as hidden states or ground truth parameters, which would otherwise require extensive exploration to uncover. Chen et al. [23] and subsequent works [24, 25, 26, 27] proposed training a student policy to imitate the behavior of a privileged teacher policy using partial observations. Del Aguila Ferrandis et al. [28] use a privileged policy to generate data for learning a state estimator, and Yu et al. [29] use for a property estimator; others incorporate privileged information into auxiliary losses [30, 31] or the critic network [32, 30]. While most existing work in these settings focuses on identifying hidden states [28, 24, 25, 26, 27, 31] or parameters [25, 26, 29] from passive observation during task execution, our work is the first to address the challenge of uncovering parameters, specifically physical property parameters in our case, that require active exploration motions different from task motions.

Method	One-shot task	Immediate task execution	Adaptive exploration time	Overlapping action space
BayesSim [12]	✗	✗	✗	✓
AdaptSim [17]	✗	✓	✗	✓
IRP [18]	✗	✓	✗	✓
DISaM [21]	✗	✓	✓	✗
Task-Oriented [9]	✓	✗	✗	✓
ASID [1]	✓	✗	✗	✓
Ours	✓	✓	✓	✓

Table 1: **One-shot task** must succeed in a single attempt. **Immediate task execution** after exploration, without simulator queries or policy retraining. **Adaptive exploration time** automatically adjusted for sufficient but minimal exploration. **Overlapping action space** shared between exploration and task execution.

4 Method

We propose a task-informed exploration approach. The core idea is to leverage the privileged task policy to automatically generate rewards based on its sensitivity to errors in estimated properties. These rewards guide the training of the exploration policy to identify task-relevant properties, while simultaneously training the online property and uncertainty estimator. At deployment, the robot begins with the exploration policy, uses uncertainty estimates to transition to task execution, and immediately executes the task policy after exploration, adjusting its motion based on the final property estimates from the exploration. Appx. A.1 outlines a detailed overview of the proposed method.

4.1 Problem Formulation

We formulate the problem as a family of Markov Decision Processes (MDPs) where the dynamics of the environment depends on the physical properties $\phi \in \Phi$ of the manipulated object. We assume that ϕ remains constant throughout the episode. Each instance takes the form of the tuple $M(\phi) = (S, A, P_\phi, R, \gamma)$, with a finite horizon H, where S is the state space, A is the action space, P_ϕ is the transition model conditioned on ϕ, R is the reward function and $\gamma \in [0, 1)$ is the discount factor. The objective is to learn a policy π that maximizes the expected discounted return across different values of the physical properties ϕ: $\max_\pi \mathbb{E}_{\phi \sim p(\phi), \pi, P_\phi} \left[\sum_{t=0}^{H-1} \gamma^t R(s_t, a_t, s_{t+1}) \right]$. The physical properties ϕ of the object are hidden, and the robot must infer them for successful task execution. Since the one-shot nature of the task precludes online adaptation or retrying the task, the robot must first explore the object to infer these properties and then adjust its task execution accordingly.

4.2 Privileged Task Policy Learning

We first train a privileged task policy π_{task} using RL with access to ground truth physical property parameters ϕ^* in simulation. We hypothesize that leveraging this privileged information reduces the need for extensive exploration and prevents convergence to suboptimal policies, leading to a higher success rate by adapting to different properties.

Task policy. During training, the task policy receives the ground truth physical property values ϕ^* as input, along with the current state observation s_t and the goal g in goal-conditioned settings, and computes actions a_t at each timestep. We define the task policy reward, r_{task}, based on the task's objectives, with positive rewards for achieving desired behaviors, such as reaching a target, and negative rewards for undesirable ones, such as violating constraints or workspace boundaries. We randomize the initial state to enable safe task execution in scenes perturbed by exploration.

4.3 Learning of Exploration and Property Estimation

While privileged access to ground truth physical property values aids task policy training, these values are unavailable in real-world settings. To address this, we simultaneously train (1) an exploration policy π_{exp} to perform motions that are informative for estimating the physical properties ϕ prior to task policy execution, and (2) an online property estimator f_ϕ to infer these properties from state observations during exploration. This estimator allows immediate task execution after exploration—unlike prior works [9, 1] that require querying a simulator for offline inference.

Exploration policy. Next, we train an exploration policy π_{exp} that computes actions a_t to obtain observations informative for identifying the physical properties of the manipulated object, given the current states s_t of the robot and the object through RL. We define the exploration policy reward as

$$r_{\text{exp}} = \begin{cases} r_{\text{estimation}} & \text{if } \forall j \; \varepsilon_{\text{estimation},j} < \varepsilon_{\text{threshold},j} \\ r_{\text{failure}} & \text{otherwise} \end{cases}, \tag{1}$$

where $\varepsilon_{\text{estimation},j}$ represents the estimation error for the j-th physical property, and $\varepsilon_{\text{threshold},j}$ is the threshold for the j-th property. The robot receives a positive reward $r_{\text{estimation}}$ if the estimation errors of all properties are below their respective thresholds. We obtain the estimation errors $\varepsilon_{\text{estimation}}$ by

computing the difference between the ground truth physical property values ϕ^* and the estimated physical property values $\hat{\phi}$ estimated by the physical property estimator f_ϕ, as

$$\varepsilon_{\text{estimation},j} = \left| \phi_j^* - \hat{\phi}_j \right|. \tag{2}$$

We adopt on-policy RL for stable training under non-stationary rewards caused by dependence on the simultaneously trained estimator. The robot receives a negative reward r_{failure} for violating the workspace boundaries, ensuring feasible task execution after exploration, without manual reset.

Online physical property estimator. For online physical property estimation during exploration, we employ a Long Short-Term Memory (LSTM) [33], as temporal information is essential for capturing object dynamics (see Appx. C.1 for alternative approaches, *e.g.,* a Transformer-based estimator). Between each training update, the exploration policy collects a new dataset $\mathcal{D}$ of rollouts, consisting of states s_t and the corresponding ground truth physical property values ϕ^*. At each update of the exploration policy, we also update the estimator by minimizing the estimation loss $\mathcal{L}_{est}$ (see Appx. A.2 for details) using the dataset $\mathcal{D}$. This simultaneous training of the estimator on observations from the most recent exploration policy mitigates out-of-distribution issues, a known challenge in learned estimators [19] and forward models [1, 34].

4.4 Task-Informed Exploration Reward Design

The exploration policy reward function in Eq. (1) requires specifying the estimation error thresholds $\varepsilon_{\text{threshold},j}$. While humans can leverage intuition and physics knowledge to identify which parameters are relevant to the target task, manually specifying optimal threshold values for multiple properties is non-intuitive. Moreover, tuning these thresholds for each task is cumbersome, as the relevance of each physical property varies across tasks. To address this, we propose a method to automatically generate these estimation thresholds by modeling task performance sensitivity to estimation error in each property. Since we compute these thresholds from task performance sensitivity, these exploration rewards serve as a surrogate for the task reward. This reward design enables the robot to learn exploratory motions that lead to high task performance, prioritizing the identification of task-relevant properties—without executing the task policy during exploration policy training.

Modeling task performance sensitivity. For each physical property, we assume a uni-modal relationship between the task success rate y and the estimation error ε, where task performance is highest when the estimation is accurate and decreases as the error increases. The rate at which task success deteriorates with increasing error reflects the sensitivity of the task to each property.

To model this relationship, we fit a parametric uni-modal function $g_j(\varepsilon)$ to empirical data $\mathcal{D}_{\varepsilon,j}$ for each j-th physical property, where g represents any uni-modal function. The dataset $\mathcal{D}_{\varepsilon,j}$ consists of pairs of estimation error levels ε and the corresponding task success rates y achieved by the task policy. During data collection, we roll out the privileged task policy, systematically replacing the ground truth value of the j-th physical property with perturbed values at varying levels of estimation error. We model the sensitivity of each property individually by introducing errors into one property at a time while keeping all others unperturbed, assuming that the impact of each property on task performance is independent of the others.

Computing task-informed exploration rewards. From the fitted uni-modal function g_j, we compute a set of estimation error thresholds $\varepsilon_{\text{threshold},j}$ for each physical property j, such that the task success rate remains above a proportion p of the maximum success rate achieved by the privileged task policy. Specifically, we solve:

$$g_j(\varepsilon_{\text{threshold},j}) \geq p \cdot \max_{\varepsilon} g_j(\varepsilon) \tag{3}$$

for each property. These estimation error thresholds define the success criteria in the exploration reward in Eq. (1). Properties that are more relevant to the task result in tighter thresholds, encouraging the exploration policy to estimate those properties with higher accuracy.

4.5 Uncertainty-Based Policy Switching

In the training phase, we define exploration success using estimation error, as in Eq. (1). However, this error is inaccessible at test time on a physical setup. To transition from exploration to task execution once the property estimates are sufficiently accurate, we 1) estimate the uncertainty of the property estimates, and 2) determine the uncertainty thresholds required for successful task execution by modeling the relationship between each property's uncertainty and the task outcome.

Uncertainty estimator. We estimate the predictive uncertainty of the physical property estimator f_ϕ using an ensemble approach that captures both aleatoric uncertainty (from data noise) and epistemic uncertainty (from model limitations) [35]. We define f_ϕ as an ensemble of M neural networks, indexed by i, each comprising two heads: one for the predicted mean $\hat{\phi}_i(s_t)$ (denoted $\hat{\phi}_{i,t}$) and another for the predicted covariance $\hat{\Sigma}_i(s_t)$ (denoted $\hat{\Sigma}_{i,t}$) [36, 37, 38]. Assuming a heteroscedastic setting, i.e. the uncertainty depends on the models' input, where $p(\phi|s_t) = \mathcal{N}(\hat{\phi}_{i,t}, \hat{\Sigma}_{i,t})$, we train each model i to minimize the negative log-likelihood. The ensemble consists of an equally weighted mixture of Gaussians which, for simplicity, we approximate with a single Gaussian distribution, such that $p(\phi|s_t) = \mathcal{N}(\hat{\phi}_t, \hat{\Sigma}_t)$, where the mean and covariance are those of the mixture [37]:

$$\hat{\phi}_t = \frac{1}{M}\sum_{i=1}^{M}\hat{\phi}_{i,t}, \qquad \hat{\Sigma}_t = \underbrace{\frac{1}{M}\sum_{i=1}^{M}\hat{\Sigma}_{i,t}}_{\substack{\text{mean of the}\\ \text{individual covariances}}} + \underbrace{\frac{1}{M}\sum_{i=1}^{M}\left(\hat{\phi}_{i,t}\hat{\phi}_{i,t}^T\right) - \hat{\phi}_t\hat{\phi}_t^T}_{\substack{\text{covariance of the}\\ \text{mixture means}}}.$$

We use the mean $\hat{\phi}_t$ as the predicted physical property values, and the covariance $\hat{\Sigma}_t$ as the measure for predictive uncertainty, where the mean of the individual covariances captures aleatoric uncertainty, while the covariance of the mixture means captures epistemic uncertainty.

Computing uncertainty thresholds. Finally, we compute the uncertainty thresholds required for successful task execution. We roll out the exploration and task policies to collect uncertainty data labeled with task outcomes (success or failure). Then, we calculate the p-th percentile of the uncertainty values for each property from successful task trials (*i.e.,* $q\%$ of successful trials have uncertainty values lower than this threshold). Therefore, the task policy is likely to succeed when the uncertainty values are below these thresholds. These thresholds enable the robot to assess exploration success and switch to the task policy without direct access to estimation error during testing.

5 Experiments

We evaluate our task-informed exploration approach in simulation and on a physical robot. We provide additional implementation details in Appx. B.

Tasks. We evaluate our method on three tasks: `Striking` a puck with unknown physical properties toward an unreachable goal; `Edge Pushing` a box with unknown contents to the table edge, where incorrect property estimates cause it to fall; and the classic `CartPole` task. Task details are in Appx. B.1.

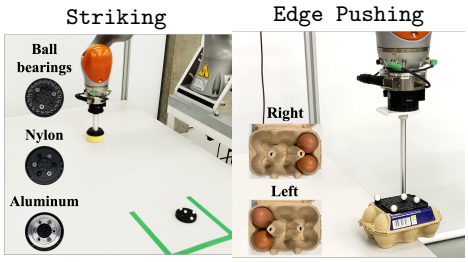


Figure 2: Manipulation tasks.

Baselines. We compare our method with the following baselines: Domain Randomization (DR) [10], which randomizes physical properties for generalization; DR+Stack, which augments DR with stacked observations to capture the temporal dynamics of objects influenced by physical properties; DR+LSTM, which integrates an LSTM into PPO to model temporal dependencies, similar to dynamic randomization with LSTM [13] but using PPO instead of RDPG; DR+Stack+Est, which extends DR+Stack with one head for action computation and another for property estimation; Student [23], which trains a student policy using DAgger; RMA [39], which trains to encode observations by regressing towards the privileged latent representation of physical properties; and UP-OSI [29], which simultaneously trains a privileged policy and an online property estimator.

5.1 Does the task-informed exploration approach improve task performance?

Ours. Fig. 3 summarizes the results for the one-shot `Striking` task. Our method achieves a 90.1% success rate, significantly outperforming all baselines, which reach at most 40%. Our policies achieve 92.3% success in exploration (see Appx. C.1) and 98.7% in task, demonstrating that the exploration policy estimates properties with sufficient accuracy for successful task performance. We confirmed consistent results on the `Edge Pushing` task (see Appx. C.2).

DR baselines. DR alone learns an average motion across properties and achieves only a 25.4% success rate. In methods with temporal information, DR+Stack and DR+LSTM, exploratory pushing motions emerge but attains only 35.4% and 23.3% task success, respectively. These methods rely on delayed task rewards, provided only after task execution, to evaluate the effectiveness of the exploration, making it difficult to associate the exploratory motions with the rewards. Adding rewards for property estimation, as in DR+Stack+Est, also achieves only 34.8% due to the challenge of balancing exploration and task execution rewards within a single policy.

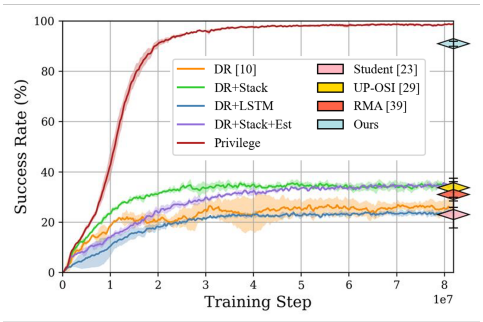


Figure 3: Performance of different methods on the `Striking` task, with mean and standard deviation reported across three training seeds.

Privileged baselines. The methods leveraging a privileged policy (Student, UP-OSI, and RMA) achieved success rates of 23.0%, 33.7%, and 31.0%, respectively. Since the privileged task policy lacks exploratory behavior, imitating it (Student) or rolling it out—whether using online estimation (UP-OSI) or latent encodings (RMA)—leads to task failure.

5.2 Do task-informed exploration rewards capture task-relevant physical properties?

Fig. 4 presents task sensitivity to errors in each physical property parameter. In the `Striking` task, the CoM in the y-direction and dynamic friction show a sharper decline in success rate as error increases, compared to less sensitive parameters such as static friction. This sensitivity model indicates that the former parameters are more relevant for performing the task, resulting in tighter estimation thresholds for exploration rewards. The results, aligning with our physics-based intuition, demonstrate that our method automatically identifies task-relevant properties. Similarly, in the `Edge Pushing` and `CartPole` tasks, each property exhibits distinct relationships, reflecting varying levels of relevance to task performance. See Appx. C.4 for further analysis on all tasks.

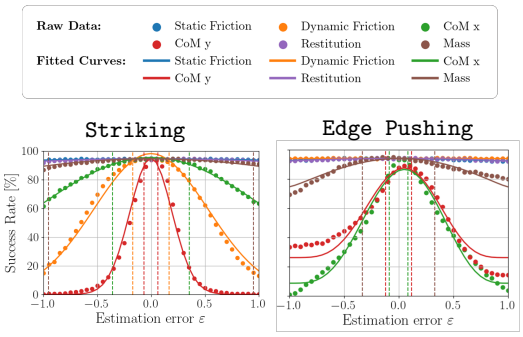


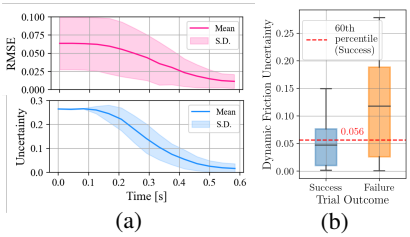
Figure 4: Uni-modal functions fitted to the relationship between task success rate and normalized property estimation errors, modeling task sensitivity. Dashed lines indicate the computed estimation error thresholds.

5.3 How do task-informed exploration rewards and uncertainty estimates contribute?

Task-informed exploration rewards. We compare the task-informed rewards from Eq. (3) against task-agnostic rewards on the `Striking` task. When thresholds are uniformly too high (*i.e.*, $\epsilon = \max_j \epsilon_{\text{threshold},j}$), task success drops to 47.3%, failing to accurately estimate task-relevant properties. When thresholds are uniformly too low (*i.e.*, $\epsilon = \min_j \epsilon_{\text{threshold},j}$), exploration success drops to 8.0%, requiring highly accurate identification of all properties. In contrast, task-informed thresholds achieve 92.3% exploration success and 90.1% task success, supporting our hypothesis that comput-

ing the highest estimation thresholds (to trigger exploration rewards) while maintaining sufficiently low thresholds (to obtain more accurate estimates) is crucial. See Appx. C.3 for complete results.

Uncertainty estimates. Fig. 5a shows that both estimation errors and uncertainties decrease as exploration progresses when rolling out the exploration policy 100 times for the `Striking` task. Further, Fig. 5b shows that low uncertainty in task-relevant property estimates leads to task success in sim2sim transfer (*i.e.,* , rolling out in PyBullet). The dotted red line indicates the computed uncertainty threshold for policy switching. See Appx. C.6 for a full analysis, including uncertainty estimates of task-irrelevant properties. These findings support using uncertainty as a proxy for estimation error in physical setups and for determining when to switch to task execution.



(a) (b)

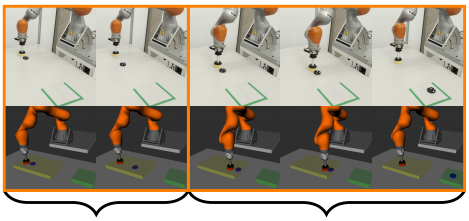
Figure 5: (a) RMSE and uncertainty of dynamic friction over time. (b) Uncertainty distribution for successful vs. failed trials.

5.4 How does our approach perform on a physical robot?

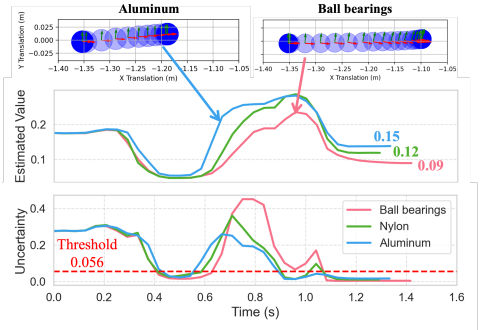
We deploy our method on the robot, with examples provided in the video and on the website. Our approach addresses sim-to-real challenges by learning to explore and estimate task-relevant physical properties of objects, and by enabling the task policy to adjust its motion based on the estimated properties.

On the `Striking` task, we evaluated on pucks with three distinct friction properties: ball bearings, nylon, and aluminum. During exploration, each puck exhibits distinct motion patterns due to differences in dynamic friction, as shown in Fig. 6. The online property estimator successfully capture these differences, with dynamic friction estimates converging to 0.09 for ball bearings, 0.12 for nylon, and 0.15 for aluminum, as the uncertainty drops.

Given the estimated properties, our method achieves 8/9 successful runs across three trials per puck. These results demonstrate that the learned exploration policy and online property estimator provide accurate property estimates, and the uncertainty estimates enable a policy switching, leading to one-shot task success. We provide detailed results for the `Striking` and `Edge Pushing` tasks in Appx. D.



(a) Top: The robot's exploration (poking the puck to infer friction and CoM) and task execution (striking to the target). Bottom: RViz view with the workspace (yellow) and target (green).



(b) Estimated dynamic friction and its uncertainty during exploration. The policy switches when all property uncertainties fall below their thresholds, though we only show dynamic friction here.

Figure 6: Robot experiments on `Striking` task.

6 Conclusion

We present a task-informed exploration RL approach that enables robots to perform exploratory motions for identifying task-relevant properties. Our approach automatically generates task-informed exploration rewards by modeling the sensitivity of a privileged task policy to estimation errors in each property. Additionally, we introduce an uncertainty-based mechanism to transition from exploration to task execution once the property estimates achieve sufficient accuracy for a given task. We evaluate our approach on the `Striking` and the `Edge Pushing` tasks with objects of varying friction and center of mass, demonstrating significantly improved task performance over alternative methods. Our analysis shows that the exploration rewards capture the relative importance of physical properties in two manipulation tasks and the classical CartPole example. Finally, we validated our approach in a physical setup, showing that the robot successfully performs exploration followed by task execution using property and uncertainty estimates.

Limitations

Correlated properties for exploration reward design. Our method for obtaining the task-informed exploration rewards, by fitting uni-modal functions independently for each property, assumes uncorrelated effects of the properties' estimation errors on the reduction of task performance. The correlated case would require designing and testing more complex reward functions.

Geometry and complex dynamics. In this work, we assumed a fixed shaped object. Generalizing over different shapes requires finding suitable input representations for the observed shape when learning the privileged task policy. Extending our approach to represent more complex dynamics—such as non-uniform friction, object deformation, or viscosity, poses the same representation challenge due to the high-dimensionality of the parameter space. Additionally, simulation of such dynamics is computationally expensive, making it demanding for learning policies using RL.

Sim2real model mismatch. Despite our method explicitly handling the sim2real transfer by learning to explore and estimate the physical property parameters of the object, we still observed distinct behavior in the highly dynamic striking task when the puck had a shifted center of mass, indicating a significant sim2real mismatch in the dynamic model itself. While our method can handle sim-to-real gaps due to parameter mismatch, it is unable to handle mismatches in the dynamic model itself, as it learns parameter estimators in simulation.

Sensor modalities. Finally, in our experiments, we use only kinematic observations to infer property parameters. It remains unexplored how to best utilize other sensing modalities, such as force-torque, tactile, audio, and vision to identify a wider range of properties and adapt to more diverse tasks and environments.

Acknowledgments

This work is supported by the JST Moonshot R&D (Grant No. JPMJMS2031). We appreciate all reviewers for the valuable feedback.

References

[1] M. Memmel, A. Wagenmaker, C. Zhu, D. Fox, and A. Gupta. ASID: Active exploration for system identification in robotic manipulation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=jNR6s6OSBT.

[2] M. Y. Aoyama, J. Moura, N. Saito, and S. Vijayakumar. Few-shot learning of force-based motions from demonstration through pre-training of haptic representation. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi:10.1109/icra57147.2024.10610502.

[3] N. Saito, N. B. Dai, T. Ogata, H. Mori, and S. Sugano. Real-time liquid pouring motion generation: End-to-end sensorimotor coordination for unknown liquid dynamics trained with deep neural networks. In *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019. doi:10.1109/ROBIO49542.2019.8961718.

[4] T. Lopez-Guevara, R. Pucci, N. K. Taylor, M. U. Gutmann, S. Ramamoorthy, and K. Subr. Stir to pour: Efficient calibration of liquid properties for pouring actions. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. doi:10.1109/IROS45743.2020.9340852.

[5] S. J. Lederman and R. L. Klatzky. Hand movements: A window into haptic object recognition. *Cognitive Psychology*, 19(3):342–368, 1987. doi:https://doi.org/10.1016/0010-0285(87)90008-9.

[6] Y.-L. Tai, Y. C. Chiu, Y.-W. Chao, and Y.-T. Chen. SCONE: A food scooping robot learning framework with active perception. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=yHlUVHWnBN.

[7] R. Antonova, J. Yang, P. Sundaresan, D. Fox, F. Ramos, and J. Bohg. A bayesian treatment of real-to-sim for deformable object manipulation. *IEEE Robotics and Automation Letters*, 7(3):5819–5826, 2022. doi:10.1109/lra.2022.3157377.

[8] N. Saito, T. Ogata, S. Funabashi, H. Mori, and S. Sugano. How to select and use tools? : Active perception of target objects using multimodal deep learning. *IEEE Robotics and Automation Letters*, 6(2):2517–2524, 2021. doi:10.1109/LRA.2021.3062004.

[9] J. Liang, S. Saxena, and O. Kroemer. Learning active task-oriented exploration policies for bridging the sim-to-real gap. In *Robotics: Science and Systems XVI*, 2020. doi:10.15607/rss.2020.xvi.085.

[10] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2017. doi:10.1109/iros.2017.8202133.

[11] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. doi:10.48550/arXiv.1910.07113.

[12] F. Ramos, R. Possas, and D. Fox. Bayessim: Adaptive domain randomization via probabilistic inference for robotics simulators. In *Robotics: Science and Systems XV*, 2019. doi:10.15607/rss.2019.xv.029.

[13] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018. doi:10.1109/icra.2018.8460528.

[14] L. Ljung. *System Identification: Theory for the User*. Prentice Hall information and system sciences series. Prentice Hall PTR, 1999. ISBN 9780136566953. URL https://books.google.co.uk/books?id=nHFoQgAACAAJ.

[15] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg. Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022. doi:10.1109/ICRA46639.2022.9811651.

[16] S. Kolev and E. Todorov. Physically consistent state estimation and system identification for contacts. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015. doi:10.1109/HUMANOIDS.2015.7363481.

[17] A. Z. Ren, H. Dai, B. Burchfiel, and A. Majumdar. Adaptsim: Task-driven simulation adaptation for sim-to-real transfer. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=9GRE34K0SB.

[18] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song. Iterative residual policy: For goal-conditioned dynamic manipulation of deformable objects. *The International Journal of Robotics Research*, 43(4):389–404, 2024. doi:10.1177/02783649231201201.

[19] K. N. Kumar, I. Essa, S. Ha, and C. K. Liu. Estimating mass distribution of articulated objects using non-prehensile manipulation. *Object Representations for Learning and Reasoning Workshop NeurIPS 2020*, 2020. URL https://orlrworkshop.github.io/program/orlr_25.html.

[20] W. Zhou, L. Pinto, and A. Gupta. Environment probing interaction policies. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryl8-3AcFX.

[21] S. Dass, J. Hu, B. Abbatematteo, P. Stone, and R. Martín-Martín. Learning to look: Seeking information for decision making via policy factorization. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=B2X57y37kC.

[22] D. Pechyony and V. Vapnik. On the theory of learnining with privileged information. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper_files/paper/2010/file/c73dfe6c630edb4c1692db67c510f65c-Paper.pdf.

[23] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, 2020. URL https://proceedings.mlr.press/v100/chen20a.html.

[24] Y. Fuchioka, C. C. Beltran-Hernandez, H. Nguyen, and M. Hamaya. Robotic object insertion with a soft wrist through sim-to-real privileged training. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. doi:10.1109/iros58592.2024.10801575.

[25] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47):eabc5986, 2020. doi:10.1126/scirobotics.abc5986.

[26] H. Qi, B. Yi, S. Suresh, M. Lambeta, Y. Ma, R. Calandra, and J. Malik. General in-hand object rotation with vision and touch. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=RN0OjfIV-X.

[27] X. Lin, Y. Wang, Z. Huang, and D. Held. Learning visible connectivity dynamics for cloth smoothing. In *Conference on Robot Learning*, 2022. URL https://proceedings.mlr.press/v164/lin22a.html.

[28] J. Del Aguila Ferrandis, J. Moura, and S. Vijayakumar. Learning visuotactile estimation and control for non-prehensile manipulation under occlusions. In *8th Annual Conference on Robot Learning*, 2024. URL https://openreview.net/forum?id=oSU7M7MK6B.

[29] W. Yu, J. Tan, C. Karen Liu, and G. Turk. Preparing for the unknown: Learning a universal policy with online system identification. In *Robotics: Science and Systems XIII*, 2017. doi:10.15607/rss.2017.xiii.048.

[30] E. S. Hu, J. Springer, O. Rybkin, and D. Jayaraman. Privileged sensing scaffolds reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=EpVe8jAjdx.

[31] L. Röstel, J. Pitz, L. Sievers, and B. Bäuml. Estimator-coupled reinforcement learning for robust purely tactile in-hand manipulation. In *2023 IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*, 2023. doi:10.1109/humanoids57100.2023.10375194.

[32] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. *Robotics: Science and Systems XIV*, Jun 2018. doi:10.15607/rss.2018.xiv.008.

[33] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[34] P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. In *International conference on machine learning*, pages 5779–5788, 2019.

[35] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023. doi:10.1007/s10462-023-10562-9.

[36] D. Nix and A. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, volume 1, pages 55–60 vol.1, 1994. doi:10.1109/ICNN.1994.374138.

[37] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[38] R. L. Russell and C. Reale. Multivariate uncertainty in deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7937–7943, 2021. doi:10.1109/TNNLS.2021.3086757.

[39] A. Kumar, Z. Fu, D. Pathak, and J. Malik. Rma: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems XVII*, 2021. doi:10.15607/rss.2021.xvii.011.

[40] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021. URL https://github.com/isaac-sim/IsaacGymEnvs.

[41] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, A. Mandlekar, B. Babich, G. State, M. Hutter, and A. Garg. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023. doi:10.1109/LRA.2023.3270034.

[42] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. doi:10.48550/arXiv.1707.06347.

Appendix

A Method

A.1 Task-informed exploration framework

Our proposed approach utilizes the task policy π_{task} to autonomously discover task-informed exploration rewards for training the exploration policy π_{exp}, without manual design. These rewards enable us to learn an exploration policy that generates informative motions to estimate the physical properties relevant to the task. Additionally, we compute the uncertainty of the physical property estimates and use them as a criterion to switch from the exploration to the task phase, as a surrogate of a estimation error. The task policy then uses the last estimate of the physical properties $\hat{\phi}$ from the exploration phase as input. Fig. 7 outlines the proposed method.

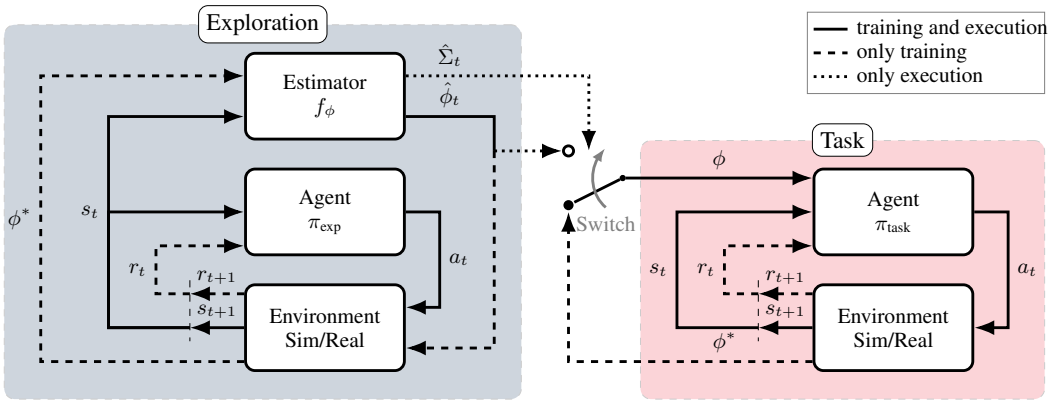


Figure 7: Method outline. The exploration component involves simultaneous training of an exploration policy π_{exp} and a properties estimator f_ϕ, given ground truth property labels ϕ^*, and executing the policy until the uncertainty $\hat{\Sigma}_t$ gets lower than a given threshold. The task component entails training a task policy, given the privileged information of the ground truth properties ϕ^*, which during execution takes the last property estimate $\hat{\phi}$ of the exploration phase as an input.

A.2 Training the physical property and uncertainty estimator

In the property estimator ensemble f_ϕ, each neural network i outputs a predicted mean $\hat{\phi}_{i,t}$ and covariance $\hat{\Sigma}_{i,t}$, given the current state s_t. To train each network, we model the conditional distribution of the true physical properties as a multivariate Gaussian parameterized by the predicted mean and covariance, i.e. $p(\phi|s_t) = \mathcal{N}(\hat{\phi}_{i,t}, \hat{\Sigma}_{i,t})$. Therefore, we use the corresponding negative log-likelihood (up to an additive constant) as the loss criterion:

$$\mathcal{L}_{est} = \frac{1}{2} \ln \left(\left| \hat{\Sigma}_{i,t} \right| \right) + \frac{1}{2} \left(\phi^* - \hat{\phi}_{i,t} \right)^T \hat{\Sigma}_{i,t}^{-1} \left(\phi^* - \hat{\phi}_{i,t} \right). \tag{4}$$

B Experimental setup

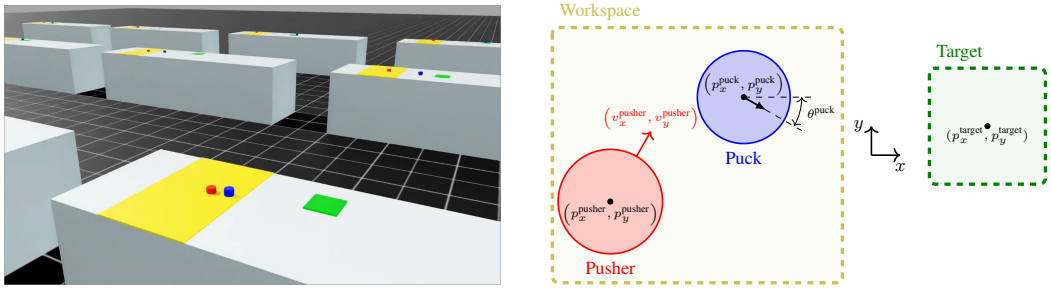
B.1 Tasks

Striking. The goal of this task is to strike a puck into a given target area beyond the robot's reachable workspace. Fig. 8a shows the simulated environment, and Fig. 8b provides a schematic representation, highlighting the non-overlapping yellow workspace and green target area. This highly dynamic task exemplifies the scenario where the robot must adjust its motion based on the physical properties of the object and is unable to recover from failure once the puck becomes unreachable, motivating the need to perform exploratory motions before executing the main task. For each trial, the puck and the robot start from a fixed position, and the task is successful if the puck arrives within a

$20\,\text{cm} \times 20\,\text{cm}$ square at the goal target coordinates $(p_x^{\text{target}}, p_y^{\text{target}})$. The state s_t consists of the puck's position $(p_x^{\text{puck}}, p_y^{\text{puck}})$ and orientation θ_{puck}, and the robot's pusher position $(p_x^{\text{pusher}}, p_y^{\text{pusher}})$, and the action a_t consists of the pusher velocity $(v_x^{\text{pusher}}, v_y^{\text{pusher}})$. The properties ϕ contain: static friction, dynamic friction, restitution, center of mass (CoM) in the x and y directions, and mass.

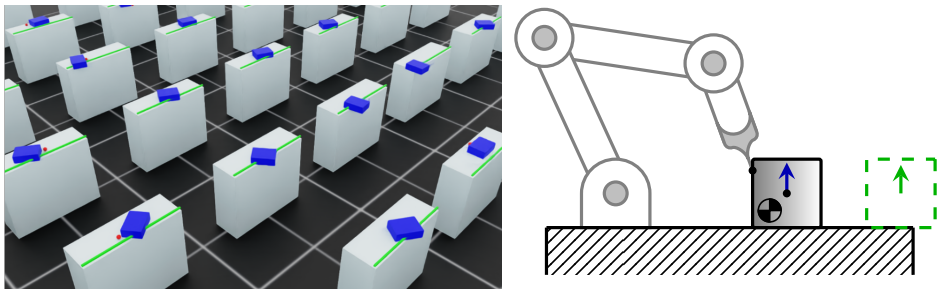
Edge Pushing. This task uses the same setup, randomized physical properties, state, and action definitions as the `Striking` task. The goal is to push a box with unknown contents to a target position at the edge of the table. Inaccurate estimates of the box's physical properties can lead to catastrophic and irreversible failure—specifically, the box falling off the table. Fig. 9a shows the simulated environment, and Fig. 9b illustrates the green target position. The task succeeds if the box arrives within $10, \text{cm}$ of the target coordinates $(p_x^{\text{target}}, p_y^{\text{target}})$ without falling off the table.

CartPole. Additionally, we evaluate our approach for modeling task sensitivity to estimation error on the classic CartPole benchmark example from Isaac Lab [40] to assess its generalizability to tasks involving a different set of physical properties. Fig. 10a shows the simulated environment, and Fig. 10b provides a schematic representation. For each trial, the pole starts from an angle sampled from $\mathcal{U}[-0.25, 0.25]\,\text{rad}$. The task is successful if the pole remains upright within a tolerance of $\pi/2\,\text{rad}$ for $5\,\text{s}$. The state consists of the current cart position p_x, cart velocity $\dot{p}_x$, pole angle θ, and pole angular velocity $\dot{\theta}$, and the action a_t specifies the force τ applied to the cart along the x direction. The properties ϕ contain: cart joint friction, pole joint friction, cart mass, and pole mass.



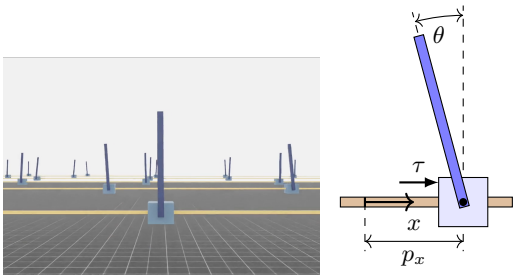
(a) (b)

Figure 8: `Striking` environment (a) in Isaac Lab [40, 41] and (b) top view schematics. The red cylindrical pusher, on the left of the dark blue puck, moves at a given velocity inside of the yellow reachable workspace. The goal of the task is to strike the puck to reach the square green target area.



(a) (b)

Figure 9: `Edge Pushing` environment (a) in Isaac Lab [40, 41] and (b) lateral view schematics.



(a) (b)

Figure 10: `CartPole` environment (a) in Isaac Lab [40, 41] and (b) horizontal view schematics. The square cart slides on the horizontal slider when pushed with a force τ. The goal of the task is to balance the rectangular pole vertically, *i.e.*, $\theta = 0$.

B.2 Simulation setup

We develop Striking and CartPole simulation environments using Isaac Lab [40, 41], as depicted in Fig. 8a and Fig. 10a. We use an NVIDIA GeForce RTX 2080 SUPER GPU and NVIDIA GeForce RTX 4090 throughout the experiments.

Striking. We abstract the robot as a cylindrical pusher to accelerate training, and model the puck as a fixed-size cylinder. We randomize the puck's physical properties ϕ, by sampling from the distributions specified in Table 2. The environment operates at a control frequency of 24 Hz, with a maximum episode duration of $H = 300$ steps or 12.5 seconds. Furthermore, we utilize PyBullet [42] to evaluate the performance of our uncertainty estimation by rolling out the policy and estimator on a different physics engine from the one used for training.

Edge Pushing. This environment uses the same setup as `Striking`, with the puck replaced by a fixed-size box.

CartPole. We randomize the CartPole properties ϕ, by sampling from the distributions specified in Table 2. The environment operates at a control frequency of 60 Hz, with a maximum episode duration of $H = 300$ steps or 5 seconds.

B.3 Training setup

Privileged task policy. We train the privileged task policy π_{task} using Proximal Policy Optimization (PPO) [43] with a continuous action space, and stack the last 5 states, $\{s_t, s_{t-1}, \ldots, s_{t-4}\}$, to capture temporal information. For the `Striking` and `Edge Pushing` tasks, the policy function consists of a neural network architecture with four linear layers of sizes 256, 128, 64, and 2, followed by Tanh non-linearities, and the value function consists of the same architecture, but with the final linear layer (1) that outputs the state value prediction. Table 3 provides a summary of the key hyperparameters and training settings. For the `CartPole` example we use the default training settings.

Task rewards. For the `Striking` and `Edge Pushing` tasks, the robot receives a positive reward of $r_t = +30$ for positioning the object (the puck or the box) within the target area and a penalty of $r_t = -15$ if either the object or the robot's end effector violates the workspace boundaries. For the `CartPole` task, we define the reward function as $r_t = +1$ when the pole stays within $\pi/2$ rad of the upright position and the cart remains within the track limits. Violating either condition results in $r_t = -2$ and immediate episode termination as a failure.

Exploration policy. We also use PPO to train the exploration policy π_{exp}, consisting of an LSTM layer with 128 hidden units, followed by three fully connected layers with 256, 64, and 2 units, and

Task	Physical Property	Distribution
`Striking`	Static Friction	$\mathcal{U}[0.05, 0.3]$
	Dynamic Friction	$\mathcal{U}[0.05, 0.3]$
	Restitution	$\mathcal{U}[0.0, 1.0]$
	Mass (kg)	$\mathcal{U}[0.02, 0.5]$
	Center of Mass Distance (m)	$\mathcal{U}[0, 0.7 \times$ puck radius$]$
	Center of Mass Angle (rad)	$\mathcal{U}[0, 2\pi]$
`Edge Pushing`	Static Friction	$\mathcal{U}[0.05, 0.3]$
	Dynamic Friction	$\mathcal{U}[0.05, 0.3]$
	Restitution	$\mathcal{U}[0.0, 1.0]$
	Mass (kg)	$\mathcal{U}[0.02, 0.5]$
	Center of Mass Distance (m)	$\mathcal{U}[0, 0.7 \times$ box width$]$
	Center of Mass Angle (rad)	$\mathcal{U}[0, 2\pi]$
`CartPole`	Cart Joint Friction	$\mathcal{U}[0.0, 1.0]$
	Pole Joint Friction	$\mathcal{U}[0.0, 1.0]$
	Cart Mass (kg)	$\mathcal{U}[0.1, 10.0]$
	Pole Mass (kg)	$\mathcal{U}[0.1, 10.0]$

Table 2: Randomization range of physical properties.

Tanh non-linearities. The value function consists of the same architecture, but with the final linear layer (1) that outputs the state value prediction. Table 3 summarizes the key hyperparameters.

Physical property and uncertainty estimator. Each network in the ensemble consists of two LSTM layers with 128 hidden units, followed by a fully connected layer that outputs $2 \times \|\phi\|$ units, where $\|\phi\|$ is the dimensionality of the physical properties being estimated. The first $\|\phi\|$ units, to which we apply a Tanh nonlinearity, correspond to the predicted *mean* of the physical properties $\hat{\phi}_{i,t}$. The remaining $\|\phi\|$ units correspond to the predicted natural logarithm of the diagonal elements of the *covariance* matrix $\hat{\Sigma}_{i,t}$. We use the natural logarithm to improve numerical stability during training and assume a diagonal covariance matrix to simplify parametrisation and reduce the dimensionality of the output. To form the ensemble, we train five networks with different random initialization weights.

Noise. We add two types of noise to the observations: step-wise noise, applied independently at each timestep to simulate sensor noise, and episodic noise, which remains constant throughout the episode to simulate calibration errors. Table 4 summarizes the distributions of the observation noise.

Hyperparameter	Value
Parallel Environments	8192
Initial Learning Rate (α)	3×10^{-4}
Optimizer	Adam
Batch Size	4096
Rollout Steps	Task: 16, Exploration: 64
Number of Epochs	8
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.95
Clip Range (ϵ)	0.2
Entropy Coefficient (β)	0.0
Total Timesteps	24,000

Table 3: Key hyperparameters used for PPO training of the task and exploration policies.

Task	Input	Noise Distribution	
		Step	Episode
`Striking/ Edge Pushing`	Object position (m)	$\mathcal{N}(0, 0.0025^2)$	$\mathcal{N}(0, 0.0025^2)$
	Object orientation (rad)	$\mathcal{N}(0, 0.01^2)$	$\mathcal{N}(0, 0.01^2)$
	Pusher position (m)	$\mathcal{N}(0, 0.0025^2)$	$\mathcal{N}(0, 0.0025^2)$
`CartPole`	Pole joint position (rad)	$\mathcal{N}(0, 0.15^2)$	$\mathcal{N}(0, 0.15^2)$
	Pole joint velocity ($\mathrm{rad/s}$)	$\mathcal{N}(0, 0.15^2)$	$\mathcal{N}(0, 0.15^2)$
	Cart joint position (m)	$\mathcal{N}(0, 0.05^2)$	$\mathcal{N}(0, 0.05^2)$
	Cart joint velocity ($\mathrm{m/s}$)	$\mathcal{N}(0, 0.05^2)$	$\mathcal{N}(0, 0.05^2)$

Table 4: Summary of step-wise noise, simulating sensor noise at each timestep, and episodic noise, simulating calibration and systematic errors across the entire episode, for the Striking and CartPole tasks.

B.4 Hardware setup

We evaluate our approach on a physical setup using a KUKA iiwa robot arm equipped with a pusher extension, as shown in Fig. 11. We use a Vicon motion capture system to track the object's current pose, and map the policy actions (*i.e.,* pusher velocities) to robot joint configurations using inverse kinematics. We evaluate two hardware tasks: `Striking` a puck with three friction levels and three center-of-mass locations (Fig. 11), and `Edge Pushing` a box of eggs placed on one of two sides (Fig. 2).

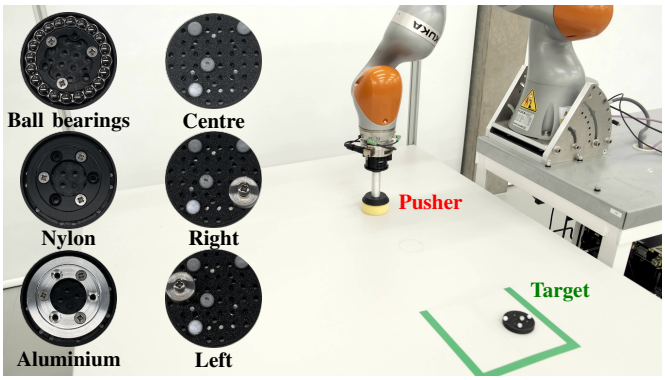


Figure 11: The **task-informed exploration** approach enables the robot to autonomously learn how to explore and identify the physical properties of objects relevant to a given task. For one-shot tasks such as striking, the robot must first identify the object's properties through exploratory motions to achieve the task success.

C Additional results

C.1 Exploration policy training — Striking

Simultaneous training of exploration policy and property estimator. Fig. 12 shows the training performance of the exploration policy π_{exp}, trained using task-informed exploration rewards and the estimator loss defined in Eq. (4), for the striking task. The results show the average over three random seeds, with shaded regions indicating the standard deviation. We demonstrate that, despite the exploration rewards being non-stationary due to their dependence on the simultaneously trained estimator, on-policy RL achieves stable training and consistently converges to an exploration success rate above 90% across seeds.

Alternative system identification approaches. Our framework supports alternative system identification modules, as long as they can output both estimates and uncertainty. While the LSTM-based property estimator is sufficient for our short exploration horizon, we also tested a Transformer-based estimator, achieving similar exploration success (Fig. 13). Our method can accommodate such models for tasks requiring longer temporal dependencies during exploration or multi-modal inputs. Additionally, although we chose a learned property estimator for fast, online inference and ease of integration into an RL pipeline without additional simulation queries, investigating alternative modules (*e.g.,* sim-in-the-loop approaches) is promising future work.

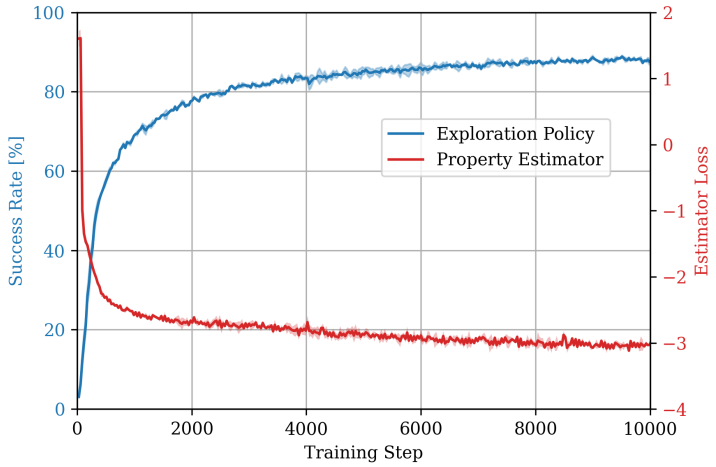


Figure 12: Exploration policy training and estimator loss of LSTM-based estimator.

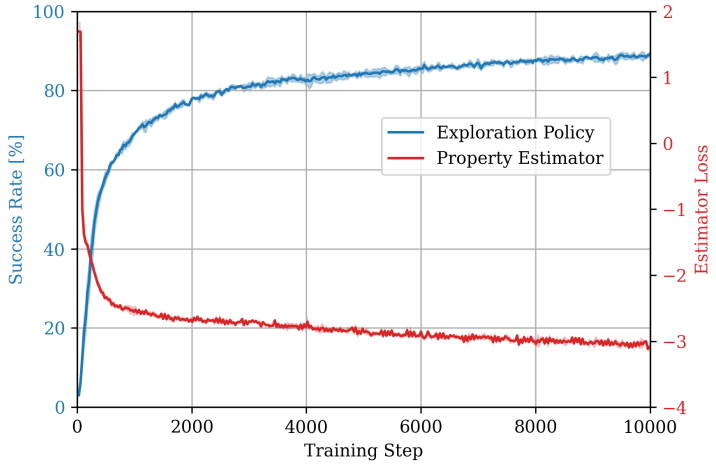


Figure 13: Exploration policy training and estimator loss of Transformer-based estimator.

C.2 Edge Pushing baseline comparison — Edge Pushing

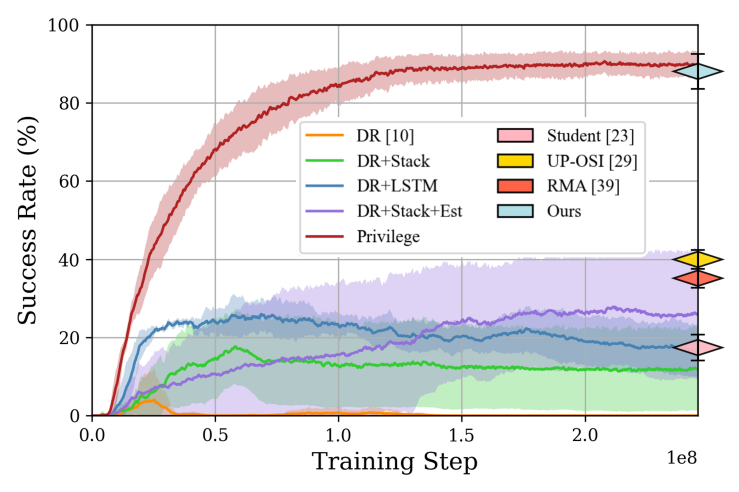


Figure 14: Performance with different training configurations of the control policy for Edge Pushing task. We report mean and standard deviation across five training seeds.

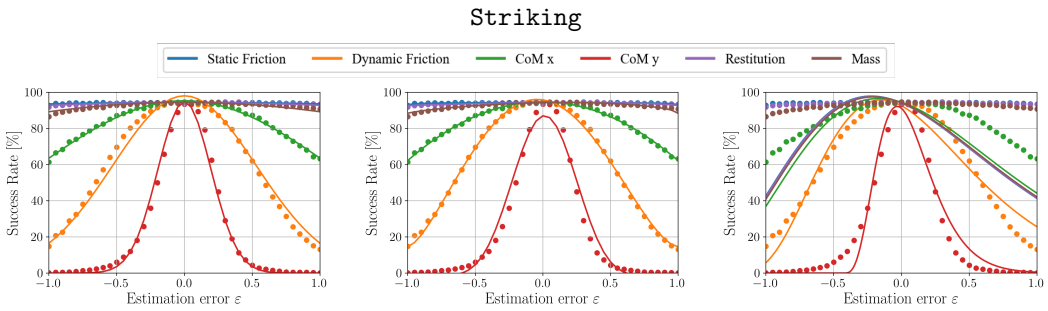
C.3 Task-agnostic vs. task-informed exploration rewards

Method	Success Rate (%)	
	Exploration	Task
Task-agnostic Min.	8.0	2.9
Task-agnostic Max.	91.7	47.3
Task-informed	92.3	90.1

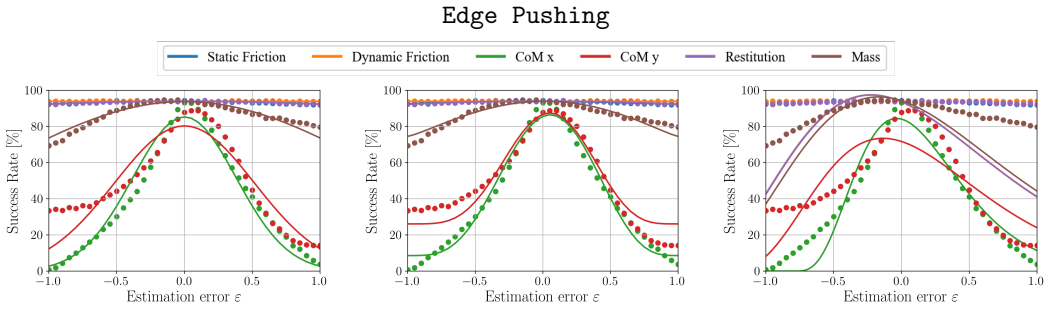
Table 5: Ablation on task-agnostic vs. task-informed exploration rewards. We provide detailed results in Appx. C.5.

C.4 Sensitivity analysis — Striking, Edge Pushing, CartPole

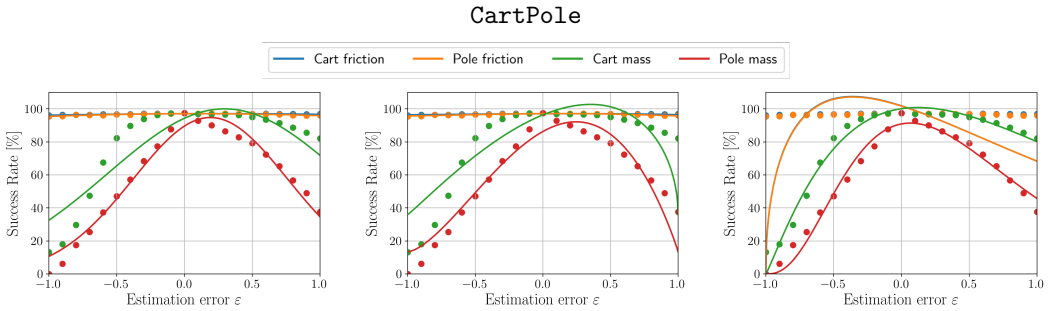
While our approaches can apply any uni-modal functions, we applied three different uni-modal distributions, the Gaussian, Beta, and Gamma distributions, to model the sensitivity of each property to each task.



(a) Gaussian function (b) Beta function (c) Gamma function



(d) Gaussian function (e) Beta function (f) Gamma function



(g) Gaussian function (h) Beta function (i) Gamma function

(j) Different uni-modal functions fitted to model the sensitivity of the task success rate to the estimation error on each property for the `Striking`, `Edge Pushing`, and `CartPole` tasks.

C.5 Estimation error during exploration policy training

Fig. 16 illustrates how the Root Mean Squared Error (RMSE) of the property estimates at the end of the exploration episode evolves during training. We report mean and standard deviation (SD) across the training environments. With the task-informed exploration policy, the estimation error for relevant physical properties, such as CoM and dynamic friction, decreases, while the error for less relevant properties remains high. By the end of training, the estimation errors for all properties fall below the thresholds indicated by the dotted lines. In contrast, task-agnostic exploration policies result in high estimation errors across all physical properties. The exploration reward with high estimation thresholds for all properties (Task-agnostic (Max.)) proves non-informative. Due to the high thresholds, even large estimation errors fall below them, failing to incentivize accurate property estimation. On the other hand, setting thresholds too low for all properties (Task-agnostic (Min.)) makes it difficult to achieve the required thresholds, leading to sparse rewards. Additionally, training becomes more challenging, as it requires learning more complex behaviors to accurately estimate all properties.

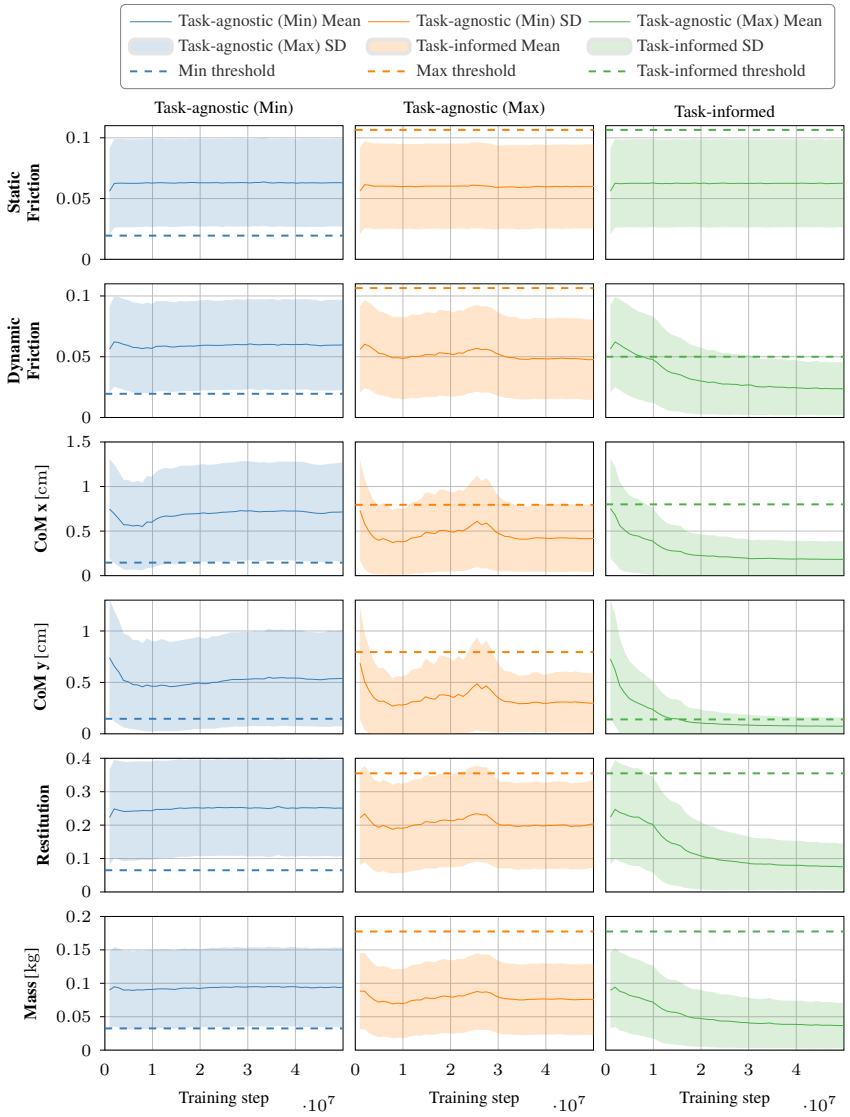


Figure 16: RMSE of the property parameter estimation at the end of the episode during training (solid line) and the estimation threshold (dashed line).

C.6 Can we use uncertainty for policy switching?

Relationship between uncertainty and estimation error. First, we evaluate whether the estimated uncertainty reflects the estimation errors by rolling out the exploration policy 100 times in Isaac Lab. Fig. 17 shows the mean and standard deviation of the estimation errors and uncertainties over the first 14 timesteps ($\approx$ 0.6 seconds) of the exploration episodes. The plots demonstrate that as the exploration progresses, both the estimation errors and uncertainties decrease for task-relevant properties, such as the CoM in the y-direction and dynamic friction. This result suggests that the uncertainty estimates effectively reflect the estimation error, supporting our approach of using uncertainty as a surrogate when the estimation error is inaccessible at test time.

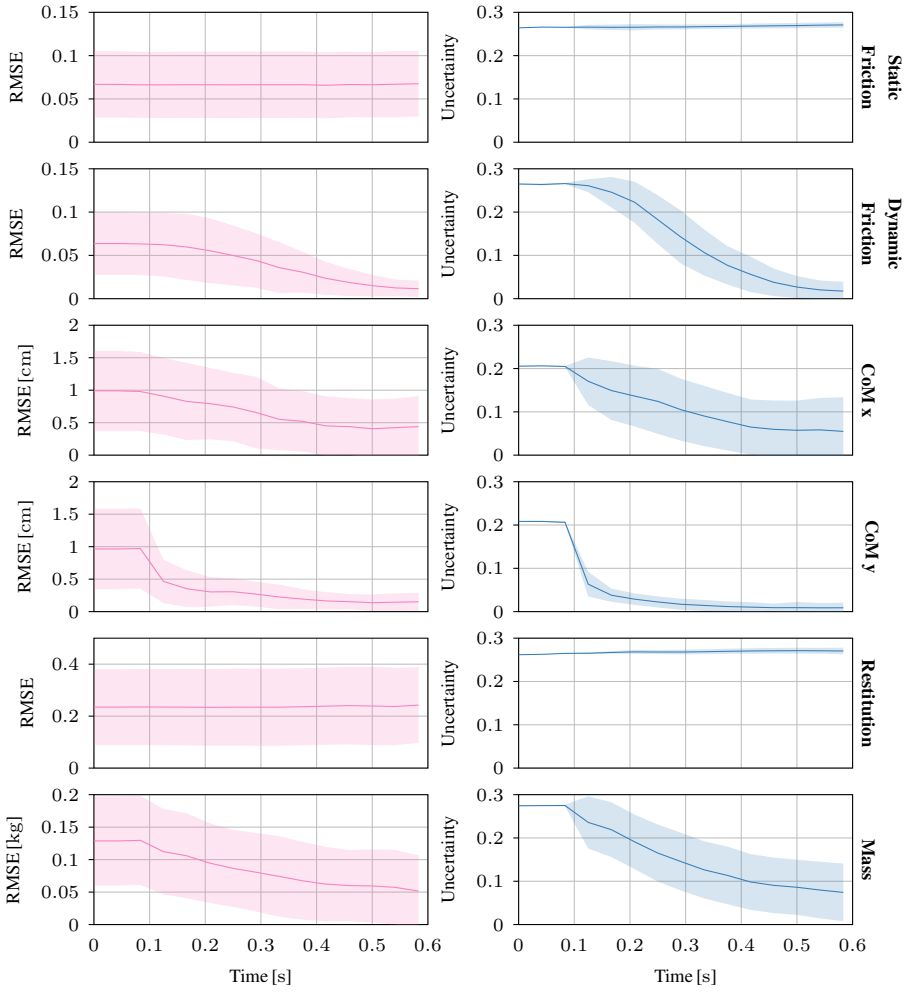


Figure 17: Estimation errors and uncertainties of property parameters during the exploration episode. As the exploration progresses, both the estimation errors and uncertainties decrease for task-relevant properties.

Relationship between uncertainty and task success. Next, we evaluate the relationship between uncertainty estimates and task success. For this evaluation, we roll out the task policy 100 times in PyBullet using estimates from the exploration policy, running for the maximum episode length. We use a simulator different from the one used for training to better assess uncertainty when there is a domain gap between training and testing, as we intend to use these uncertainties in a physical setup. Fig. 18 presents box plots of uncertainty at the end of the exploration episode for successful and

failed trials. The plots show that low uncertainty in task-relevant property estimates leads to task success, while high uncertainty results in failure. These findings suggest that uncertainty estimates during exploration predicts task outcomes and indicates when property estimates are sufficiently accurate to transition from exploration to the task phase.

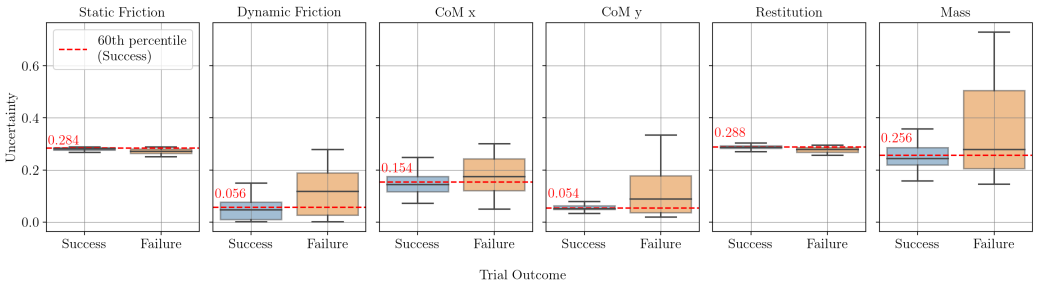


Figure 18: Relationship between uncertainties and task outcomes when rolling out the exploration and task policies in PyBullet. Low uncertainties lead to task success, while high uncertainties correlate with failures, particularly for task-relevant properties such as CoM in the y-direction and dynamic friction.

D Robot experiments

D.1 Striking

Varying surface friction. We provide dynamic friction estimates and its uncertainty for all trials on pucks with different levels of friction. Fig. 19 shows that the dynamic friction estimates consistently converge to 0.09 for ball bearings, 0.12 for nylon, and 0.15 for aluminum, with uncertainty dropping below 0.056, the uncertainty thresholds computed in Fig. 18.

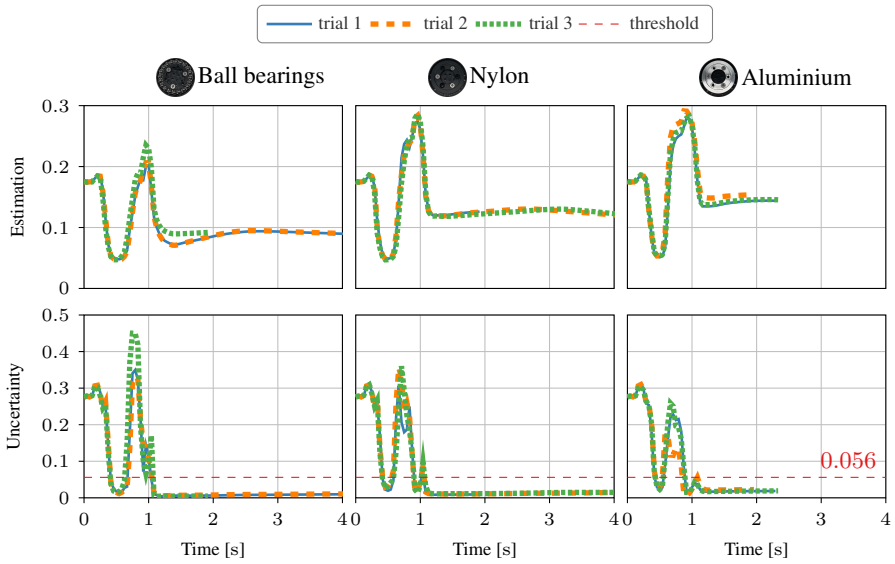


Figure 19: Estimates of **dynamic friction** and its associated uncertainty during exploration policy execution on the physical setup. Dynamic friction estimates consistently converge to 0.9, 0.12, and 0.15 for the pucks with ball bearings, nylon, and aluminum, respectively.

Shifted center of mass. We also evaluated pucks with three varying locations of the center of mass, as shown in the second column in Fig. 11. When the CoM is in the center, the robot successfully

estimates its location and completes the task, achieving 3/3 successful runs. However, for pucks with shifted CoM—either to the left or right—exploration fails due to inaccurate estimates and consistently high uncertainty, as shown in Fig. 20. We observe that, while shifts in the CoM primarily cause spinning with minimal trajectory deviation in the Isaac Lab training environment, they result in more pronounced trajectory divergence in the physical setup. This discrepancy likely arises because the simulation models the puck-table contact as a point contact, whereas real-world contact involves full surface contact, altering the friction distribution. Bridging this sim-to-real gap by modeling such contact in simulation or utilizing a small amount of real-world data to learn unmodeled dynamics is a promising direction for future work.

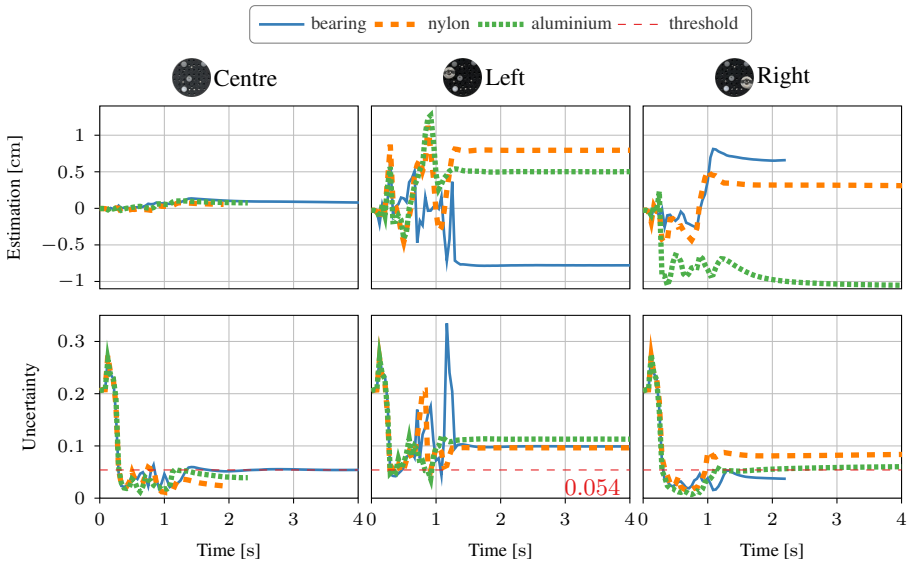


Figure 20: Estimates of the y component of the **center of mass** and its associated uncertainty during exploration policy execution on the physical setup. Uncertainty decreases in successful trials, while it remains high (*i.e.,* above the threshold 0.054 obtained from Fig. 18) in failed trials.

D.2 Edge Pushing

Shifted center of mass. For the `Edge Pushing` task, we tested the robot with two different center-of-mass (CoM) locations by placing the eggs on either side of the egg cartons. When we shifted the CoM to the left, the robot estimated its location 5.8cm off-center to the left, achieving 5/5 successful runs. When we shifted the CoM to the right, the robot estimated its location 5.8cm off-center to the right, achieving 3/5 successful runs. Fig. 21 shows that the robot's CoM estimates converge to the correct values, with uncertainty dropping below 0.018—the computed threshold.

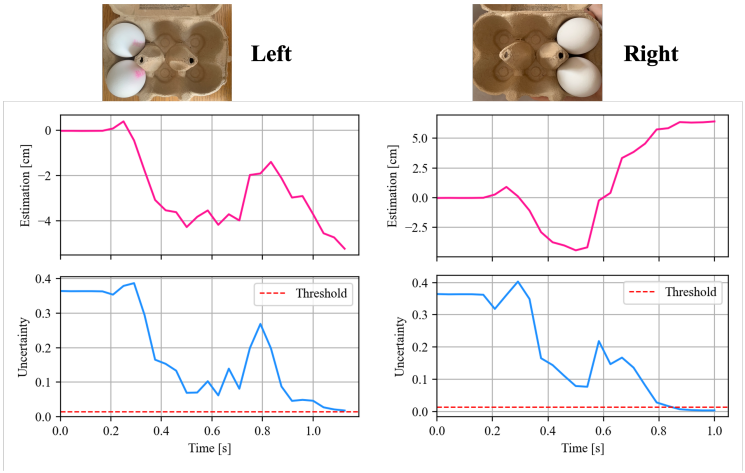


Figure 21: Estimates of the x component of the **center of mass** and its associated uncertainty during exploration policy execution on the physical setup for the `Edge Pushing` task. Uncertainty decreases below the computed threshold 0.018, leading to successful trials .