CAIMAN: Causal Action Influence Detection for Sample-Efficient Loco-Manipulation

Yuanchen Yuan[1] Jin Cheng[2], Núria Armengol Urpí[2], Stelian Coros[2]
[1]Department of Mechanical and Process Engineering
[2]Department of Computer Science
ETH Zurich
cassidity19@gmail.com,{jin.cheng,nuria.armengolurpi,stelian.coros}@ethz.ch

Abstract: Enabling legged robots to perform non-prehensile loco-manipulation is crucial for enhancing their versatility. However, learning behaviors such as whole-body object pushing often necessitates sophisticated planning strategies or extensive task-specific reward shaping. In this work, we present CAIMAN, a practical reinforcement learning framework that encourages the agent to gain *control* over other entities in the environment. CAIMAN leverages causal action influence as an intrinsic motivation objective, allowing legged robots to efficiently acquire object pushing skills even under sparse task rewards. We employ a hierarchical control strategy, combining a low-level locomotion module with a high-level policy that generates task-relevant velocity commands and is trained to maximize the intrinsic reward. To estimate causal action influence, we learn the dynamics of the environment by integrating a kinematic prior with data collected during training. We empirically demonstrate CAIMAN's superior sample efficiency and adaptability to diverse scenarios in simulation, as well as its successful transfer to real-world systems without further fine-tuning.

Keywords: Reinforcement Learning, Loco-manipulation, Intrinsic Motivation

1 Introduction

Modern legged robots continue to impress with their diverse capabilities, from traversing challenging terrains [1, 2, 3] to executing agile maneuvers such as backflips and parkour [4, 5, 6]. As expectations for autonomous robots rise, enabling these systems to interact with their environments remains an open research challenge [7, 8, 9]. A common approach is to equip legged robots with external manipulators for prehensile manipulation [10, 11, 12, 13]. Although effective, these methods are often constrained by the size and weight of objects they can successfully manipulate. In contrast, non-prehensile manipulation demands whole-body coordination and remains a non-trivial task.

Traditional approaches for whole-body manipulation explicitly model robot-object interactions, often relying on complex planning and optimization for coordinating motion [14]. These methods require accurate models of both the robot and the environment, restricting their scalability for high-dimensional systems with complex, stochastic dynamics, and limiting contact to predefined regions [15, 16]. Learning-based methods offer a more scalable alternative, improving computational efficiency and reducing dependence on precise object estimation [17, 18, 19]. However, these methods frequently rely on tedious reward shaping [20] or learning curriculums to guide exploration and foster meaningful behavior. Furthermore, the large exploration space of loco-manipulation tasks often necessitates special treatment, such as learning from behavioral priors [21, 22, 23] or task-agnostic explorative rewards [24, 25], to encourage meaningful engagement with the environment.

To alleviate these challenges, we introduce CAIMAN, a framework for training non-prehensile object pushing skills in legged robots. CAIMAN employs a hierarchical control structure that decou-

Workshop on Resource-Rational Robot Learning at CoRL 2025.

ples high-level planning from low-level locomotion. We train robust locomotion using an existing pipeline, and we learn the high-level policy with a simple yet effective reward structure consisting of only three terms: a sparse task reward, an action regularizer, and an intrinsically motivated explorative reward. The sparse task reward provides a general signal for various pushing tasks, where the robot is only rewarded for completing the task successfully. The exploratory reward incentivizes the robot to explore and gain control over the environment via Causal Action Influence (CAI) [26], which is a measure quantifying how much influence an agent has on the states of other entities and is computed based on environment dynamics. Instead of learning the dynamics from scratch, CAIMAN combines a simple predefined kinematic prior with learned residual dynamics that compensate for physical interactions beyond the prior model, allowing accurate dynamics to be learned efficiently. We show that under a sparse task reward setting, CAIMAN achieves strong learning performance and superior sample efficiency against other baselines, including in complex scenarios with obstacles. We also demonstrate that the learned residual accurately captures complex robot-object interactions that are not modeled by the kinematic prior. Finally, we successfully transfer the learned policy to a real quadruped robot, enabling its performance of whole-body object pushing in different scenarios.

To summarize, our contribution is three-fold: 1) a general hierarchical framework for learning whole-body object pushing with legged robots in various scenarios, including navigating through obstacles; 2) an intrinsically motivated reward based on causal influence calculated from a combination of a kinematics prior and learned residual dynamics; 3) successful hardware validation on a real-world quadruped robot.

2 Related Work

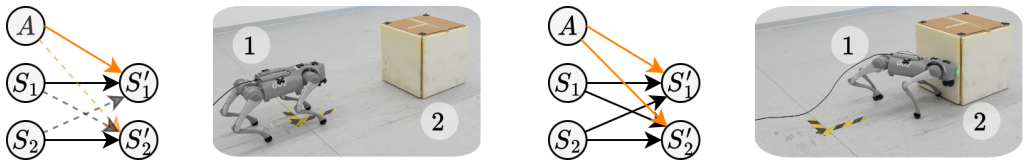
2.1 Loco-manipulation on Legged Systems

The integration of locomotion and manipulation has gained significant attention as a promising and application-oriented research field for legged robots. Traditional model-based methods that rely on accurate representations of both robot and object to optimize trajectories [11, 14, 16, 27] have shown success in tasks such as box carrying [10, 12], but often require accurate state estimation [28, 29, 30] and struggle with scalability due to the complexity of contact modes [31, 32]. Reinforcement learning (RL) is a promising alternative that avoids explicit modeling and has been successfully applied to tasks such as soccer dribbling [33, 34], button pushing [18, 35], and door opening [19, 24, 36]. However, effective exploration may be difficult to achieve due to the large search space of robot-object interactions [17, 37]. To address exploration challenges, researchers have proposed using behavior prior [38, 39] or task-agnostic explorative rewards [24, 25] as mechanisms to guide learning. Furthermore, hierarchical frameworks [40, 41] are utilized to decompose tasks into high-level planning and low-level control, enabling effective behavior across various scenarios [42, 43].

In accordance with previous works, we also implement a hierarchical control framework for whole-body object pushing, decoupling high-level velocity planning from low-level locomotion, similarly to Jeon et al. [20]. We aim to utilize a simple reward structure, as opposed to one that requires sophisticated design effort as seen in [20], to achieve sample-efficient high-level policy training, even under complex scenarios such as object manipulation through obstacles.

2.2 Intrinsically Motivated Reinforcement Learning

Intrinsic motivation (IM) [44] plays a vital role in reinforcement learning, especially when extrinsic rewards are sparse or difficult to design. Core IM mechanisms include curiosity [45], learning progress [46], and empowerment [47], each promoting exploration and skill acquisition. Curiosity, often measured via prediction errors in learned world models [48, 49, 50], rewards agents for encountering novel states, and has been previously applied for learning loco-manipulation skills [24, 25]. Learning progress encourages agents to focus on regions in the state space with rapid improvement, supporting curriculum learning and adaptive exploration [51, 52].



(a) No influence of A on S'_2

(b) Influence of A on S'_1 and S'_2

Figure 1: Illustration of the LCM (*left*) for two different environment situations $S = s$ (*right*) in the loco-manipulation task. The LCM captures the transition from S, A to S', factorized into state components. While the global SCM is fully connected (dashed and continuous lines), the LCM $\mathcal{G}_{S=s}$ (continuous lines) is causally minimal. We are interested in detecting the presence of continuous orange arrows in the LCM, i.e. the influence of the action A on next states S'.

Our work builds on empowerment, an information-theoretic quantity defined as the channel capacity between the agent's actions and its sensory observations [47, 53, 54]. Recent work has connected IM to causal reasoning, aiming to improve sample efficiency and interpretability [55, 56, 57, 58, 59]. To encourage effective exploration, we employ causal action influence (CAI) [26], a measure of an agent's ability to influence its environment and a conceptual lower bound on empowerment.

3 Preliminaries

We model decision-making in a dynamic environment as a Markov Decision Process (MDP) [60], defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ the action space, P the transition kernel, R the reward function, and γ the discount factor. Following the principle of independent causal mechanisms [61], we assume that the world consists of interacting but independent entities. This induces a state space factorization $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_N$ for N entities, where each factor $\mathcal{S}_i$ represents the state of entity i. An MDP coupled with a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ induces a *Structural Causal Model* (SCM) [62] describing the resulting trajectory distribution.

Definition 1 (Structural Causal Model [62])**.** *A SCM is a tuple $(\mathcal{U}, \mathcal{V}, F, P^u)$, where $\mathcal{U}$ are set of exogenous variables (e.g., latent randomness) sampled from P^u, $\mathcal{V}$ are set of observed variables (e.g., states, actions, rewards), F is the set of structural functions capturing the causal relations, such that functions $f_V : \mathrm{Pa}(V) \times \mathcal{U} \to V$, with $\mathrm{Pa}(V) \subset \mathcal{V}$ denoting the set of parents of V, determine the value of endogenous variables V for each $V \in \mathcal{V}$.*

SCMs are typically visualized as directed acyclic graphs, where nodes represent variables and edges indicate causal relations, as shown in Fig. 1. In our case, the SCM captures one-step transitions with variables $\mathcal{V} = S_1, \ldots, S_N, A, S'_1, \ldots, S'_N$. Due to the Markov property and flow of time, causal dependencies exist only from (S, A) to S'. While most entity pairs may theoretically interact (i.e., $S_i/A \to S'_j$ for most i, j), interactions often become sparse when we observe a specific state configuration. We use the *Local Causal Model* (LCM) to capture such context-specific structure [63].

Definition 2 (Local Causal Model [63])**.** *Given a SCM $(\mathcal{U}, \mathcal{V}, F, P^u)$ and an observation $V = v, V \subset \mathcal{V}$, the local SCM is the SCM with $F_{V=v}$ and graph $\mathcal{G}_{do(V=v)}$ obtained by pruning edges from $\mathcal{G}_{do(V=v)}$ until it is causally minimal.*

In this work, we build on the insight that encouraging agents to gain *control* over their environment facilitates learning in loco-manipulation tasks. We do so by driving the agent toward states where it can influence other entities. We make use of a principled explicit measure of local influence, the Causal Action Influence (CAI) [26, 64] measure. CAI is a state-dependent measure of control that assesses whether an agent, through its actions, can affect other entities. Graphically, it corresponds to predicting the existence of an edge $A \to S'_j$ in the LCM $\mathcal{G}_{do(S=s)}$. This influence is measured with point-wise conditional mutual information $I(S'_j; A \mid S = s)$.

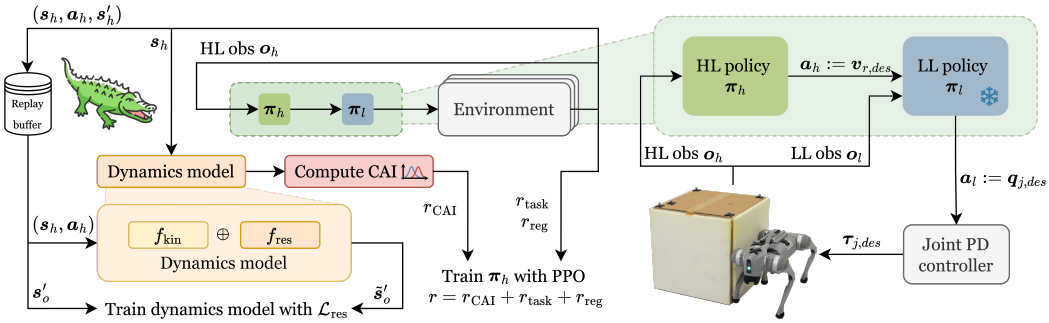


Figure 2: CAIMAN framework: The high-level (HL) policy generates desired base velocity commands, which are translated into joint commands by a low-level (LL) policy. We utilize a simple kinematic prior and learned residual dynamics to model the robot-object interaction in the environment while providing a CAI-based exploratory bonus along with the sparse task reward.

4 Method

The hierarchical framework adopted by CAIMAN is presented in Fig. 2. It consists of a high-level (HL) policy that generates desired base velocity commands and a low-level policy that translates velocity commands into joint-level actions. In this work, we introduce a novel, sample-efficient approach for learning the high-level policy, while building upon an existing pipeline [65] to train robust low-level policies.

4.1 Low-level Locomotion Policy

The low-level locomotion policy $\boldsymbol{\pi}_l$ is trained to generate target joint positions $\boldsymbol{q}_{j,des} \in \mathbb{R}^{12}$ that track a desired base velocity command $\boldsymbol{v}_{r,des} \in \mathbb{R}^3$. This command, $\boldsymbol{v}_{r,des} = (v^x_{r,des}, v^y_{r,des}, \omega^z_{r,des})$, specifies linear velocities in the longitudinal (x) and lateral (y) directions and a yaw rate, all in the robot frame. The desired joint positions $\boldsymbol{q}_{j,des}$ are tracked using joint-level proportional-derivative (PD) controllers to produce the corresponding joint torques $\boldsymbol{\tau}_{j,des} \in \mathbb{R}^{12}$. The policy is trained with velocity commands sampled uniformly from a predefined range. To ensure robust sim-to-real transfer and hardware performance during contact-rich tasks like pushing, we apply domain randomization and external perturbations following [65]. Further details of the low-level observation $\boldsymbol{o}_l$ can be found in Appendix A.

4.2 High-level Planning Policy

The high-level policy $\boldsymbol{\pi}_h$ is trained to generate desired robot velocity commands $\boldsymbol{a}_h := \boldsymbol{v}_{r,des}$ that achieve successful task completion in object pushing. Its observation $\boldsymbol{o}_h$ includes the robot's linear and angular velocities, the poses of the robot and object, the target object position $\boldsymbol{p}_t = (x_p, y_p) \in \mathbb{R}^2$, and the previous action—all expressed in the world frame. In scenarios with obstacles (e.g., walls), $\boldsymbol{o}_h$ also includes their poses in the world frame. For a detailed description of the high-level observation, refer to Appendix A.

We use Proximal Policy Optimization (PPO) [66] as the base RL algorithm, with a simple reward function composed of three terms:

$$r = w_1 \mathbb{1}_{\|\boldsymbol{p}_o - \boldsymbol{p}_t\|_2 < \epsilon} + w_2 r_{\text{CAI}} + w_3 \left\|\boldsymbol{a}_h - \boldsymbol{a}_{h,prev}\right\|_2^2, \tag{1}$$

where $\boldsymbol{p}_o$, $\boldsymbol{p}_t$ are the current and target object positions, ϵ is a success threshold, and $\boldsymbol{a}_h$, $\boldsymbol{a}_{h,prev}$ are the current and previous high-level actions, respectively. The exploration bonus r_{CAI} is derived from the CAI measure $\tilde{C}_j$ over the object j. We define the CAI reward as:

$$r_{\text{CAI}} = \tilde{C}_{j=\text{object}}(s) = \frac{1}{K} \sum_{i=1}^{K} D_{\text{KL}} \left(P_{S'_j | S=s, A=a^{(i)}} \middle\| \frac{1}{K} \sum_{k=1}^{K} P_{S'_j | S=s, A=a^{(k)}} \right), \tag{2}$$

given K actions $\{a^{(i)}\}_{i=1}^{K}$ sampled from the policy. This approximation estimates the marginal $P_{S'_j|s}$ using Monte Carlo sampling. We model the transition distribution $P_{S'_j|S=s,A=a}$ as a fixed-variance Gaussian (details in Section 4.3), which enables closed-form KL divergence calculation using the Gaussian mixture approximation [67].

The CAI reward encourages the agent to reach states where it exerts greater influence over the object, thereby promoting task-relevant exploration and learning. The weight w_2 for the CAI reward scales with the raw CAI score:

$$w_2 = w_{2,b} + \max\left(0, (r_{\text{CAI}} - \alpha_1)/\alpha_2\right), \tag{3}$$

where α_1 is the threshold for scaling and α_2 controls the scaling rate.

Finally, we foster more directed exploration by injecting time-correlated noise into the action sampling process during training [68, 69]. For more details on training the high-level policy, we refer the reader to Appendix D.

4.3 Dynamics learning

To calculate the exploration bonus r_{CAI} in (2), we learn the transition model $P_{S'_j|S=s,A=a^{(k)}}$ for all entities j. In our current setting, we focus on a single entity—the pushable object—but the framework naturally extends to multiple entities. Concretely, this reduces to learning the object transition model $P_{\boldsymbol{s}'_o|\boldsymbol{s}_h,\boldsymbol{a}_h}$. The object state $\boldsymbol{s}_o$ is defined as the object's position, while the high-level state $\boldsymbol{s}_h = (\boldsymbol{\xi}_r, \boldsymbol{\xi}_o, \boldsymbol{v}_r, \boldsymbol{v}_o)$ includes the robot's pose $\boldsymbol{\xi}_r$ and velocity $\boldsymbol{v}_r$ and the object's pose $\boldsymbol{\xi}_o$ and velocity $\boldsymbol{v}_o$. The high-level action $\boldsymbol{a}_h$ corresponds to the desired robot velocity, and the next object state $\boldsymbol{s}'_o = \boldsymbol{p}'_o = (x'_o, y'_o)$ denotes its position at the subsequent timestep. Instead of using a pretrained model, CAIMAN leverages data collected from high-level interactions during training to efficiently learn the dynamics. As described earlier, we model the object's transition probability $P_{\boldsymbol{s}'_o|\boldsymbol{s}_h,\boldsymbol{a}_h}$ as a fixed-variance Gaussian distribution $\mathcal{N}(\boldsymbol{s}'_o; f_\theta(\boldsymbol{s}_h, \boldsymbol{a}_h), \sigma^2)$, where f_θ is a neural network that predicts the mean of the distribution.

To enhance learning efficiency, we incorporate a simple kinematic prior model f_{kin}, which estimates the next object position $\tilde{\boldsymbol{p}}'_o$ using geometric reasoning based on the relative pose between robot and object and the commanded velocity. This estimate is computed by projecting the robot's velocity $\boldsymbol{a}_h$ onto the direction from the robot to the object and updating the object's position accordingly:

$$\tilde{\boldsymbol{p}}'_o = \begin{cases} \boldsymbol{p}_o + \delta t \cdot (\boldsymbol{a}_{h,xy} \cdot \delta\hat{\boldsymbol{p}}) \cdot \delta\hat{\boldsymbol{p}}, & \text{if } \|\boldsymbol{p}_o - \boldsymbol{p}_r\|_2 \le \epsilon_p \text{ and } \boldsymbol{a}_{h,xy} \cdot \delta\hat{\boldsymbol{p}} > 0 \\ \boldsymbol{p}_o, & \text{otherwise.} \end{cases} \tag{4}$$

Here, $\delta\hat{\boldsymbol{p}} = (\boldsymbol{p}_o - \boldsymbol{p}_r)/\|\boldsymbol{p}_o - \boldsymbol{p}_r\|_2$ is the unit vector pointing from the robot to the object, δt is the high-level control step size, and ϵ_p is a distance threshold. The conditions ensure that the robot is close enough to and moving toward the object.

We combine the kinematic prior with a learned residual model f_{res}, parameterized by θ, to capture complex physical interactions beyond the kinematics model, including nonlinear effects such as friction, drag, and collisions with obstacles. The final dynamics model is thus:

$$f_\theta(\boldsymbol{s}_h, \boldsymbol{a}_h) = f_{\text{kin}}(\boldsymbol{s}_h, \boldsymbol{a}_h) + f_{\text{res}}(\boldsymbol{s}_h, \boldsymbol{a}_h; \theta), \tag{5}$$

where f_{kin} is deterministic and independent of θ. We train the residual model by minimizing the mean squared error between the predicted and true object positions:

$$\mathcal{L}_{\text{res}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \left\| \tilde{\boldsymbol{s}}'^{(i)}_o - \boldsymbol{s}'^{(i)}_o \right\|_2^2, \tag{6}$$

where $\tilde{\boldsymbol{s}}'^{(i)}_o = f_{\text{kin}}(\boldsymbol{s}_h^{(i)}, \boldsymbol{a}_h^{(i)}) + f_{\text{res}}(\boldsymbol{s}_h^{(i)}, \boldsymbol{a}_h^{(i)}; \theta)$.

Although the high-level action $\boldsymbol{a}_h$ could, in principle, be sampled from the full support of the desired velocity $\boldsymbol{v}_{r,des}$ [26], not all commands are feasible for the low-level controller to execute under the robot's current state. Therefore, we define $\boldsymbol{a}_h = \tilde{\boldsymbol{v}}_r$ to be the achievable velocity, and use $\tilde{\boldsymbol{v}}_r$ for

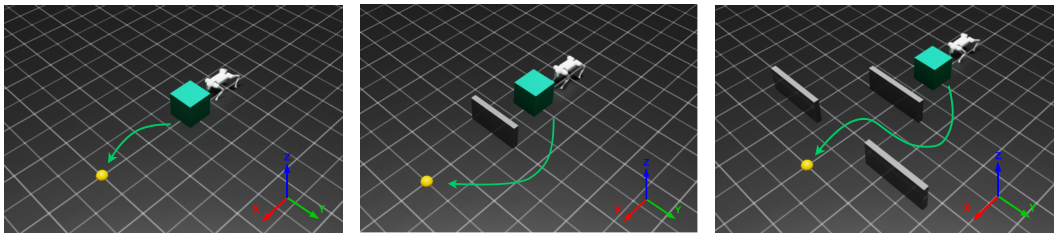


Figure 3: Illustrations for the `Single-object` (*left*), `Single-wall` (*middle*), and `Multi-wall` (*right*) tasks. The yellow sphere denotes the object's target position.

both dynamics learning and CAI computation. Given training samples $\mathcal{D} = (\boldsymbol{s}_h^{(i)}, \boldsymbol{a}_h^{(i)}, \boldsymbol{s}_h'^{(i)})$, we have access to the achieved robot velocity $\boldsymbol{v}_r'$ at the next timestep. When computing CAI as an exploration reward, we assume that the robot's velocity can only change within a limited range over one high-level step. Thus, for any state $\boldsymbol{s}_h$, we sample the action $\boldsymbol{a}_h$ from a bounded range centered on the current velocity:

$$\boldsymbol{a}_h = \tilde{\boldsymbol{v}}_r \sim \mathcal{U}[\boldsymbol{v}_r \pm \delta \boldsymbol{v}_r] \tag{7}$$

where $\delta \boldsymbol{v}_r$ is a fixed velocity deviation range. Detailed values for the velocity limits $(\delta v_r^x, \delta v_r^y, \delta \omega_r^z)$ are provided in Appendix D.

5 Experiments

We train both the high-level and low-level policies using Isaac Sim [70]. Training hyperparameters and additional details are provided in Appendix D. We evaluate CAIMAN on three pushing tasks of increasing difficulty, illustrated in Fig. 3. For the `Single-object` task, the robot must manipulate a single cuboid object to a fixed target position. The `Single-wall` task includes a fixed wall that blocks the direct path between the object and the target. The robot must navigate the object around the wall to reach the goal. Finally, the `Multi-wall` task includes multiple fixed walls that obstruct the path between the object and the target. The robot must maneuver the object through these obstacles to complete the task. The presence of fixed wall obstacles in the single-wall and multi-wall tasks introduces regions in the state space where object dynamics become more complex, reducing the robot's ability to exert consistent influence over the object. For all tasks, we train with a fixed target position. A full description of scene configurations is provided in Appendix B.

Baselines We compare our approach against several baselines. The `Heuristics` baseline trains the high-level policy using a distance-based heuristic reward, $r_{\text{heu}} = \exp\left(-\|\boldsymbol{p}_r - \boldsymbol{p}_o\|_2\right)$, encouraging the robot to minimize its distance to the object. This mirrors the main explorative reward from prior work [20]. We also conduct comparisons with other intrinsic motivation algorithms, namely Random Network Distillation (RND) [71]. We implement two variants: `RND-full`, which mirrors the original RND implementation and computes the prediction error for the full state, and `RND-object`, which computes the prediction error for the object's state only. As for ablations, we implement `CAI-kinematic`, where CAI is computed using only the kinematic prior (Eq. 4), and and `CAI-learned`, where CAI is computed using a fully learned dynamics model (no prior).

For detailed formulations of all reward functions and their coefficients, see Appendix C.

5.1 Simulation results

We evaluate the learning performance of each method by measuring the success rate for each task. A task is considered successfully completed if the distance between the object and the target is below a threshold ϵ_s at the end of an episode. The results are shown in Fig. 4.

With only sparse task rewards, task-relevant skill acquisition heavily relies on effective exploration. In the `Single-object` task, all methods—except `RND-full`—achieve some degree of success. CAIMAN, `RND-object`, and `Heuristics` perform best, while `CAI-kinematic` and `CAI-learned` underperform. The low performance of `CAI-learned` is likely due to the high

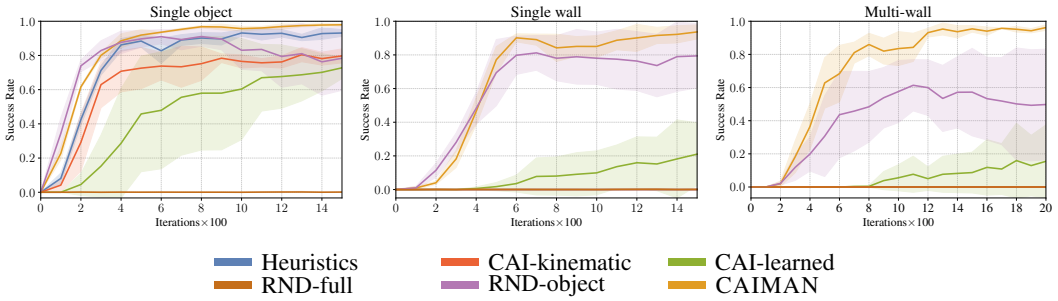


Figure 4: Policy success rate evaluated at every 100 training iterations for all methods and tasks. Results are evaluated across 800 episodes and averaged over 3 seeds, shaded area represents standard deviation.

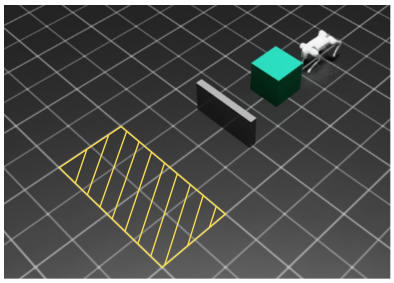


Figure 5: Single wall task with target positions randomly sampled within an area in front of the wall.

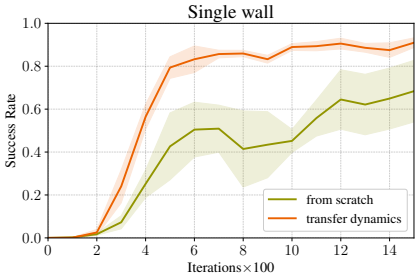


Figure 6: Learning the single wall random target task using learning from scratch and learning with models transferred from the fixed target task.

data and time demands required to learn an accurate dynamics model from scratch. In contrast, `CAI-kinematic` performs comparably to the top methods, suggesting that the naive kinematic prior is sufficient for simpler tasks with straightforward object interactions.

In the more complex `Single-wall` and `Multi-wall` tasks, only CAIMAN and `RND-object` demonstrate notable success, with `RND-object` exhibiting lower asymptotic performance than CAIMAN in both scenarios. This highlights CAIMAN's unique ability to guide task-relevant exploration even in the presence of obstacles, validating its effectiveness in cluttered environments without relying on dense extrinsic rewards. The failure of `CAI-kinematic` and `CAI-learned` in these tasks emphasizes the importance of combining a kinematic prior with a learned residual model to efficiently capture the complex object dynamics needed for effective exploration.

`RND-full` results in agents randomly exploring the state space without yielding meaningful behavior in all tasks. This finding reinforces the need for object-centric exploration bonuses when the skill to be learned involves physical interaction with objects. The performance gap between `RND-object` and CAIMAN can be partially attributed to the phenomenon of *detachment* [72], where the agent drifts away from reward-depleted areas and continuously pursues new regions with higher intrinsic reward, potentially losing avenues of task-relevant exploration.

5.2 Leveraging pretrained dynamics for new tasks

Since our framework learns the environment dynamics, we propose that the dynamics residual obtained from a previous training can be reused across different tasks—as long as the underlying dynamics remain consistent. We hypothesize that leveraging a pretrained model improves the accuracy of CAI estimates in the early stages of training, thereby enhancing the exploration signal and significantly boosting sample efficiency.

To test this hypothesis, we consider a generalization of the `Single-wall` task in which the target position is randomized instead of fixed, sampled uniformly within a predefined area as illustrated in Fig. 5. Detailed scene configurations are provided in Appendix B. We compare the original CAIMAN, which learns the dynamics residual from scratch, to one that reuses a pretrained residual

model. For the latter, we initialize the residual with one from the original `Single-wall` task and continuously fine-tune it during training with data collected from the new generalized task.

As shown in Fig. 6, the reuse of learned dynamics significantly accelerates learning with sparse rewards. The pretrained model yields more informative CAI estimates, guiding the robot toward meaningful interaction behaviors early in training. This leads to substantial gains in sample efficiency, supporting our hypothesis and demonstrating the benefits of transferring learned dynamics across related tasks.

5.3 Hardware deployment

Figure 7: Snapshots from the hardware deployment for the single object and single wall tasks. The mass of the box is 5.5 kilograms with a dimension of $(0.55, 0.55, 0.5)$ meters.

We validated our trained policies on real-world quadruped pushing tasks with Unitree Go2, as shown in Fig. 7. To bridge the sim-to-real gap, we applied domain randomization to object mass and friction during high-level policy training. Additional details are provided in Appendix E. All entities in the environment were tracked using an external motion capture system. The trained policies are deployed directly to the robot and successfully execute pushing tasks without requiring additional fine-tuning. We refer interested readers to the supplementary video for more details.

6 Conclusion

We present CAIMAN, a general framework for training whole-body pushing skills in legged robots. CAIMAN adopts a hierarchical control strategy that decouples locomotion from high-level planning. By introducing an intrinsic exploration bonus based on CAI, our method encourages the robot to gain control over relevant entities in its environment under sparse task signals—eliminating the need for tedious, hand-crafted reward shaping, even in cluttered scenarios with obstacles. To further enhance training efficiency, we bootstrap the CAI computation using a simple yet effective kinematic prior, and refine it with a learned residual dynamics model. We evaluate CAIMAN on a suite of quadruped pushing tasks and show that it consistently outperforms competitive baselines in terms of sample efficiency. We also demonstrate that the learned dynamics residual can be reused to accelerate training for new tasks, highlighting the transferability of our approach. Finally, we validate our method on real hardware, seamlessly deploying the trained policies without the need for fine-tuning. Overall, CAIMAN offers a robust and scalable solution for learning physically grounded manipulation behaviors in legged robots, paving the way toward more autonomous and adaptable robotic systems.

7 Limitations

CAI-based exploration provides meaningful guidance for robot-object interaction, but inevitably relies heavily on an accurate dynamics model. Training the high-level policy alongside the dynamics residual may result in undesired exploratory behaviors in the early stages of training when the dynamics model is still inaccurate. One way to mitigate this issue would be to leverage a pretrained world model. Our current deployment pipeline also relies on an external motion capture system,

with the incorporation of an onboard perceptive module left for future work. Another potential direction for the future is to extend CAIMAN to multi-object scenarios.

References

[1] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[2] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Science robotics*, 7(62):eabk2822, 2022.

[3] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo. Learning quadrupedal locomotion on deformable terrain. *Science Robotics*, 8(74):eade2256, 2023.

[4] B. Katz, J. Di Carlo, and S. Kim. Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 international conference on robotics and automation (ICRA)*, pages 6295–6301. IEEE, 2019.

[5] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11443–11450. IEEE, 2024.

[6] D. Hoeller, N. Rudin, D. Sako, and M. Hutter. Anymal parkour: Learning agile navigation for quadrupedal robots. *Science Robotics*, 9(88):eadi7566, 2024.

[7] S. Ha, J. Lee, M. van de Panne, Z. Xie, W. Yu, and M. Khadiv. Learning-based legged locomotion; state of the art and future perspectives. *arXiv preprint arXiv:2406.01152*, 2024.

[8] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *Annual Review of Control, Robotics, and Autonomous Systems*, 8, 2024.

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

[10] C. D. Bellicoso, K. Krämer, M. Stäuble, D. Sako, F. Jenelten, M. Bjelonic, and M. Hutter. Alma-articulated locomotion and manipulation for a torque-controllable robot. In *2019 International conference on robotics and automation (ICRA)*, pages 8477–8483. IEEE, 2019.

[11] S. Zimmermann, R. Poranne, and S. Coros. Go fetch!-dynamic grasps using boston dynamics spot with external robotic arm. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4488–4494. IEEE, 2021.

[12] J.-P. Sleiman, F. Farshidian, M. V. Minniti, and M. Hutter. A unified mpc framework for whole-body dynamic locomotion and manipulation. *IEEE Robotics and Automation Letters*, 6 (3):4688–4695, 2021.

[13] M. Liu, Z. Chen, X. Cheng, Y. Ji, R.-Z. Qiu, R. Yang, and X. Wang. Visual whole-body control for legged loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.

[14] M. Murooka, S. Nozawa, Y. Kakiuchi, K. Okada, and M. Inaba. Whole-body pushing manipulation with contact posture planning of large and heavy object for humanoid robot. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5682–5689. IEEE, 2015.

[15] E. Farnioli, M. Gabiccini, and A. Bicchi. Toward whole-body loco-manipulation: Experimental results on multi-contact interaction with the walk-man robot. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1372–1379. IEEE, 2016.

[16] A. Rigo, Y. Chen, S. K. Gupta, and Q. Nguyen. Contact optimization for non-prehensile loco-manipulation via hierarchical model predictive control. In *2023 ieee international conference on robotics and automation (icra)*, pages 9945–9951. IEEE, 2023.

[17] F. Shi, T. Homberger, J. Lee, T. Miki, M. Zhao, F. Farshidian, K. Okada, M. Inaba, and M. Hutter. Circus anymal: A quadruped learning dexterous manipulation with its limbs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2316–2323. IEEE, 2021.

[18] X. Cheng, A. Kumar, and D. Pathak. Legs as manipulator: Pushing quadrupedal agility beyond locomotion. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5106–5112. IEEE, 2023.

[19] P. Arm, M. Mittal, H. Kolvenbach, and M. Hutter. Pedipulate: Enabling manipulation skills using a quadruped robot's leg. In *41st IEEE Conference on Robotics and Automation (ICRA 2024)*, 2024.

[20] S. Jeon, M. Jung, S. Choi, B. Kim, and J. Hwangbo. Learning whole-body manipulation for quadrupedal robot. *IEEE Robotics and Automation Letters*, 9(1):699–706, 2023.

[21] H. Ha, Y. Gao, Z. Fu, J. Tan, and S. Song. Umi on legs: Making manipulation policies mobile with manipulation-centric whole-body controllers. *arXiv preprint arXiv:2407.10353*, 2024.

[22] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Demonstration-guided reinforcement learning with learned skills. *5th Conference on Robot Learning*, 2021.

[23] N. A. Urpí, M. Bagatella, O. Hilliges, G. Martius, and S. Coros. Efficient learning of high level plans from play. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10189–10196. IEEE, 2023.

[24] C. Schwarke, V. Klemm, M. Van der Boon, M. Bjelonic, and M. Hutter. Curiosity-driven learning of joint locomotion and manipulation tasks. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pages 2594–2610. PMLR, 2023.

[25] C. Zhang, W. Xiao, T. He, and G. Shi. Wococo: Learning whole-body humanoid control with sequential contacts. *arXiv preprint arXiv:2406.06005*, 2024.

[26] M. Seitzer, B. Schölkopf, and G. Martius. Causal influence detection for improving efficiency in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:22905–22918, 2021.

[27] M. P. Polverini, A. Laurenzi, E. M. Hoffman, F. Ruscelli, and N. G. Tsagarakis. Multi-contact heavy object pushing with a centaur-type humanoid robot: Planning and control for a real demonstrator. *IEEE Robotics and Automation Letters*, 5(2):859–866, 2020.

[28] J. Li and Q. Nguyen. Multi-contact mpc for dynamic loco-manipulation on humanoid robots. In *2023 American Control Conference (ACC)*, pages 1215–1220. IEEE, 2023.

[29] C. Lin, X. Liu, Y. Yang, Y. Niu, W. Yu, T. Zhang, J. Tan, B. Boots, and D. Zhao. Locoman: Advancing versatile quadrupedal dexterity with lightweight loco-manipulators. *arXiv preprint arXiv:2403.18197*, 2024.

[30] A. Schakkal, G. Bellegarda, and A. Ijspeert. Dynamic object catching with quadruped robot front legs. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6848–6855. IEEE, 2024.

[31] X. Cheng, S. Patil, Z. Temel, O. Kroemer, and M. T. Mason. Enhancing dexterity in robotic manipulation via hierarchical contact exploration. *IEEE Robotics and Automation Letters*, 9 (1):390–397, 2023.

[32] C. Smith, Y. Karayiannidis, L. Nalpantidis, X. Gratal, P. Qi, D. V. Dimarogonas, and D. Kragic. Dual arm manipulation—a survey. *Robotics and Autonomous systems*, 60(10):1340–1353, 2012.

[33] Y. Ji, G. B. Margolis, and P. Agrawal. Dribblebot: Dynamic legged manipulation in the wild. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5155–5162. IEEE, 2023.

[34] Y. Hu, K. Wen, and F. Yu. Dexdribbler: Learning dexterous soccer manipulation via dynamic supervision. *arXiv preprint arXiv:2403.14300*, 2024.

[35] Z. He, K. Lei, Y. Ze, K. Sreenath, Z. Li, and H. Xu. Learning visual quadrupedal loco-manipulation from demonstrations. *arXiv preprint arXiv:2403.20328*, 2024.

[36] M. Zhang, Y. Ma, T. Miki, and M. Hutter. Learning to open and traverse doors with a legged manipulator. *arXiv preprint arXiv:2409.04882*, 2024.

[37] Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023.

[38] J.-P. Sleiman, M. Mittal, and M. Hutter. Guided reinforcement learning for robust multi-contact loco-manipulation. In *8th Annual Conference on Robot Learning (CoRL 2024)*, 2024.

[39] F. Liu, Z. Gu, Y. Cai, Z. Zhou, S. Zhao, H. Jung, S. Ha, Y. Chen, D. Xu, and Y. Zhao. Opt2skill: Imitating dynamically-feasible whole-body trajectories for versatile humanoid loco-manipulation. *arXiv preprint arXiv:2409.20514*, 2024.

[40] A. Rigo, M. Hu, S. K. Gupta, and Q. Nguyen. Hierarchical optimization-based control for whole-body loco-manipulation of heavy objects. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15322–15328. IEEE, 2024.

[41] J. Wang, R. Dai, W. Wang, L. Rossini, F. Ruscelli, and N. Tsagarakis. Hypermotion: Learning hybrid behavior planning for autonomous loco-manipulation. In *8th Annual Conference on Robot Learning*, 2024.

[42] Y. Ji, Z. Li, Y. Sun, X. B. Peng, S. Levine, G. Berseth, and K. Sreenath. Hierarchical reinforcement learning for precise soccer shooting skills using a quadrupedal robot. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1479–1486. IEEE, 2022.

[43] K. N. Kumar, I. Essa, and S. Ha. Cascaded compositional residual learning for complex interactive behaviors. *IEEE Robotics and Automation Letters*, 8(8):4601–4608, 2023.

[44] R. Rm. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25:54–67, 2000.

[45] J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 222–227, 1991.

[46] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE transactions on autonomous mental development*, 2(3):230–247, 2010.

[47] A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 ieee congress on evolutionary computation*, volume 1, pages 128–135. IEEE, 2005.

[48] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[49] D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pages 5062–5071. PMLR, 2019.

[50] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *Seventh International Conference on Learning Representations*, pages 1–17, 2019.

[51] S. Blaes, M. Vlastelica Pogančić, J. Zhu, and G. Martius. Control what you can: Intrinsically motivated task-planning agent. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pages 1331–1340. PMLR, 2019.

[53] S. Mohamed and D. Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 28, 2015.

[54] R. Zhao, Y. Gao, P. Abbeel, V. Tresp, and W. Xu. Mutual information state intrinsic control. *arXiv preprint arXiv:2103.08107*, 2021.

[55] L. Buesing, T. Weber, Y. Zwols, S. Racaniere, A. Guez, J.-B. Lespiau, and N. Heess. Woulda, coulda, shoulda: Counterfactually-guided policy search. *arXiv preprint arXiv:1811.06272*, 2018.

[56] E. Bareinboim, A. Forney, and J. Pearl. Bandits with unobserved confounders: A causal approach. *Advances in Neural Information Processing Systems*, 28, 2015.

[57] C. Lu, B. Schölkopf, and J. M. Hernández-Lobato. Deconfounding reinforcement learning in observational settings. *arXiv preprint arXiv:1812.10576*, 2018.

[58] D. J. Rezende, I. Danihelka, G. Papamakarios, N. R. Ke, R. Jiang, T. Weber, K. Gregor, H. Merzic, F. Viola, J. Wang, et al. Causally correct partial models for reinforcement learning. *arXiv preprint arXiv:2002.02836*, 2020.

[59] S. A. Sontakke, A. Mehrjou, L. Itti, and B. Schölkopf. Causal curiosity: Rl agents discovering self-supervised experiments for causal representation learning. In *International conference on machine learning*, pages 9848–9858. PMLR, 2021.

[60] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[61] J. Peters, D. Janzing, and B. Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[62] J. Pearl. *Causality*. Cambridge university press, 2009.

[63] S. Pitis, E. Creager, and A. Garg. Counterfactual data augmentation using locally factored dynamics. *Advances in Neural Information Processing Systems*, 33:3976–3990, 2020.

[64] N. A. Urpí, M. Bagatella, M. Vlastelica, and G. Martius. Causal action influence aware counterfactual data augmentation. In *Forty-first International Conference on Machine Learning*, 2024.

[65] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. In *Conference on Robot Learning*, pages 91–100. PMLR, 2022.

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

[67] J.-L. Durrieu, J.-P. Thiran, and F. Kelly. Lower and upper bounds for approximation of the kullback-leibler divergence between gaussian mixture models. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4833–4836. Ieee, 2012.

[68] O. Eberhard, J. Hollenstein, C. Pinneri, and G. Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[69] J. Hollenstein, G. Martius, and J. Piater. Colored noise in ppo: Improved exploration and performance through correlated action sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38(11), pages 12466–12472, 2024.

[70] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, J. L. Yuan, R. Singh, Y. Guo, H. Mazhar, et al. Orbit: A unified simulation framework for interactive robot learning environments. *IEEE Robotics and Automation Letters*, 8(6):3740–3747, 2023.

[71] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[72] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995*, 2019.

A Detailed Observation Space

Below we list the observation space for low-level, high-level policies and the high-level state for calculating CAI.

Low-level observation o_l	
${}_{\mathcal{B}}\boldsymbol{v}_r \in \mathbb{R}^3$	robot linear velocity in base frame $\mathcal{B}$
${}_{\mathcal{B}}\boldsymbol{\omega}_r \in \mathbb{R}^3$	robot angular velocity in base frame $\mathcal{B}$
$\boldsymbol{q}_j \in \mathbb{R}^{12}$	Joint positions
$\dot{\boldsymbol{q}}_j \in \mathbb{R}^{12}$	Joint velocities
${}_{\mathcal{B}}\boldsymbol{g} \in \mathbb{R}^3$	Projected gravity in base frame $\mathcal{B}$
$\boldsymbol{v}_{r,des} = (v^x_{r,des}, v^y_{r,des}, \omega^z_{r,des}) \in \mathbb{R}^3$	Desired velocity command
$\boldsymbol{a}_{l,prev} \in \mathbb{R}^{12}$	Previous action
High-level observation o_h (in world frame $\mathcal{W}$)	
$\boldsymbol{v}_r \in \mathbb{R}^3$	Robot linear velocity
$\boldsymbol{\omega}_r \in \mathbb{R}^3$	Robot angular velocity
$\boldsymbol{\xi}_r = (x_r, y_r, \psi_r) \in \mathbb{R}^3$	Robot pose
$\boldsymbol{\xi}_o = (x_o, y_o, \psi_o) \in \mathbb{R}^3$	Object pose
$\boldsymbol{p}_t = (x_t, y_t) \in \mathbb{R}^2$	Target position
$\boldsymbol{a}_{h,prev} \in \mathbb{R}^3$	Previous action
additional:	
$(x_w, y_w, \psi_w) \in \mathbb{R}^3$	Wall pose (for each wall)
high-level state for CAI s_h (in world frame $\mathcal{W}$)	
$\boldsymbol{\xi}_r = (x_r, y_r, \psi_r) \in \mathbb{R}^3$	Robot pose
$\boldsymbol{\xi}_o = (x_o, y_o, \psi_o) \in \mathbb{R}^3$	Object pose
$\boldsymbol{v}_r = (v^x_r, v^y_r, \omega^z_r) \in \mathbb{R}^3$	Robot velocity
$\boldsymbol{v}_o = (v^x_o, v^y_o, v^z_o) \in \mathbb{R}^3$	Object velocity

Table A.1: Detailed observation space for each module.

B Scene Configurations

The detailed initial pose for each entity and target position are shown in Table B.1. In simulation, the object measures 0.5 meters in length, width, and height, and weighs 5.0 kilograms. The wall in the single-wall task has dimensions $(0.1, 1.0, 0.5)$ meters, while walls in the multi-wall task measure $(0.1, 1.25, 0.5)$ meters. For the transfer task of Single-wall, we sample the target position within the range of $(\mathcal{U}[3.0, 4.0], \mathcal{U}[-1.0, 1.0])$.

Task	`Single-object`	`Single-wall`	`Multi-wall`
Robot	(0.0, 0.0, $\mathcal{U}(-\pi, \pi]$)	(0.0, 0.0, $\mathcal{U}(-\pi, \pi]$)	(0.0, 0.0, $\mathcal{U}(-\pi, \pi]$)
Object	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)	(1.0, 0.0, 0.0)
Target	(3.0, 0.0)	(3.5, 0.0)	(4.0, 0.0)
Wall 1	-	(2.0, 0.0, 0.0)	(2.0, 0.0, 0.0)
Wall 2	-	-	(3.25, 1.25, 0.0)
Wall 3	-	-	(3.25, -1.25, 0.0)

Table B.1: We list the scene configurations, where the initial pose (x, y, ψ) for every existing entity and the target position (x, y) for each task (`Single-object, Single-wall, Multi-wall`) is shown.

C Rewards and Coefficients

We list the reward functions and corresponding reward parameters for all tasks and training methods in Table C.1. The RND reward term is defined as $r_{RND} = \|f(s) - \hat{f}(s)\|_2$, where s is the state to be encoded, and $f(s)$ and $\hat{f}(s)$ are the target and predictor networks, respectively.

Reward functions	
CAI	$r = w_1 r_{\text{task}} + w_2 r_{\text{CAI}} + w_3 r_{\text{reg}}$
RND	$r = w_1 r_{\text{task}} + w_4 r_{\text{RND}} + w_3 r_{\text{reg}}$
heuristics	$r = w_1 r_{\text{task}} + w_5 r_{\text{heu}} + w_3 r_{\text{reg}}$
r_{CAI} weight	Eq. 3

Reward coefficients							
Task	w_1	w_3	w_4	w_5	$w_{2,b}$	α_1	α_2
Single object	15	-5e-3	10	0.01	40	12e-5	1.5e-6
Single wall	40	-5e-3	10	0.01	40	12e-5	1.5e-6
Multi-wall	40	-5e-3	10	0.01	40	12e-5	1.5e-6

Table C.1: Simulation training reward parameters. All CAI methods (`CAI-kinematic, CAI-learned`, CAIMAN) use the CAI reward function, all RND methods (`RND-full, RND-object`) use the RND reward function, while only the `Heuristics` method uses the heuristics reward function.

D Additional training details

The low-level policy operates with a control interval of 0.02 seconds, while the high-level policy has a control interval of 0.2 seconds. For every iteration through the high-level control loop, we step the environment by 0.2 seconds, compute the rewards and terminations, collect transitions for the high-level and dynamics policies, and calculate and add the CAI exploration bonus to the rewards buffer. We trained 1500 iterations for the single object and single wall tasks and 2000 iterations for the multi-wall task, where each iteration consists of 10 high-level control steps. All tasks have an episode length of 20 seconds.

In addition to CAI, we also generate meaningful exploration through colored noise during policy rolling out [68, 69]. By sampling from time-correlated actions, we reduce the possibility of meaningless back-and-forth behavior that could result from commonly used white-noise samples. We select a correlation strength parameterized as $\beta = 0.5$, corresponding to a colored noise between white and pink.

We list all the hyperparameters used for training the high-level loco-manipulation policy and dynamics residual model in Table D.1.

E Domain randomization for hardware deployment

Table E.1 presents the environment parameters that were randomized in simulation during training policies for hardware deployment.

F Additional Experiment: Dense task reward

Though we primarily focused on a sparse task reward, we also trained all tasks in simulation under a dense task reward for all methods. The dense task reward, defined as $r_{\text{task}}^{dense} = \exp - \|\boldsymbol{p}_o - \boldsymbol{p}_t\|_2$, awards an amount inversely proportional to the euclidean distance between the object and the target. Combining the dense task rewards with a distance-based heuristic reward resembles the settings from prior work [20].

Environment hyperparameters	
number of environments	4096
$v^x_{r,des}$ (m/s)	$[-1.0, 1.0]$
$v^y_{r,des}$ (m/s)	$[-1.0, 1.0]$
$\omega^z_{r,des}$ (rad/s)	$[-1.0, 1.0]$
δv^x_r (m/s)	0.3
δv^y_r (m/s)	0.3
$\delta \omega^z_r$ (rad/s)	0.4
threshold for kinematic prior model ϵ_p (m)	0.7
threshold for sparse task reward ϵ (m)	0.1
success rate threshold ϵ_s (m)	0.15
high-level policy frequency (Hz)	5
low-level policy frequency (Hz)	50
number of sampled action in CAI K	64
PPO hyperparameters	
policy network	$[512, 256, 128]$
policy activation function	ELU
policy initial standard deviation	1.0
value network	$[512, 256, 128]$
value activation function	ELU
number of mini-batches	4
number of epochs per iteration	5
clip threshold	0.2
learning rate schedule	adaptive
desired KL divergence	0.01
value function loss coefficient	1.0
entropy coefficient	0.005
max gradient norm	1.0
γ	0.99
λ	0.95
Dynamics hyperparameters	
fixed variance σ	1.0
residual model	$[256, 128]$
batch size	4096
number of epochs per iteration	8
learning rate	0.0001
buffer size	10000000

Table D.1: High-level policy and dynamics residual training hyperparameters.

Parameter	**Range**
Object mass (kg)	$\mathcal{U}[3.5, 7.5]$
Object friction coefficient	$\mathcal{U}[0.5, 1.5]$
Initial robot position (m)	x: $\mathcal{U}[-0.1, 0.1]$, y: $\mathcal{U}[-0.1, 0.1]$
Initial object position (m)	x: $\mathcal{U}[0.9, 1.1]$, y: $\mathcal{U}[-0.1, 0.1]$

Table E.1: Randomized parameters in simulation for training HL policies for hardware.

Task performance during training is shown in Fig. F.1. For learning the single object task with dense rewards, all methods demonstrate similar learning speeds and success rates, which indicates that within simpler, less cluttered environments, a dense reward scheme sufficiently guides task learning.

However, for the dense reward single wall and multi-wall tasks, all CAI-imbued learning methods (CAIMAN, `CAI-kinematic, CAI-learned`) exhibit better sample efficiency over the heuristics method, indicating that learning the actions necessary for obstacle navigation is facilitated through

CAI exploration. CAIMAN and `CAI-kinematic` achieve the fastest convergence and the highest success rates, while CAI-learned is slightly less sample efficient. This can be attributed to the fact that learning dynamics from scratch initially produces erroneous predictions, which results in inaccurately high CAI scores and diverts the robot from acquiring task specific skills. In contrast, with only the kinematic prior model, the resulting exploration behavior is less adversely impacted when skill learning is heavily guided by dense rewards.

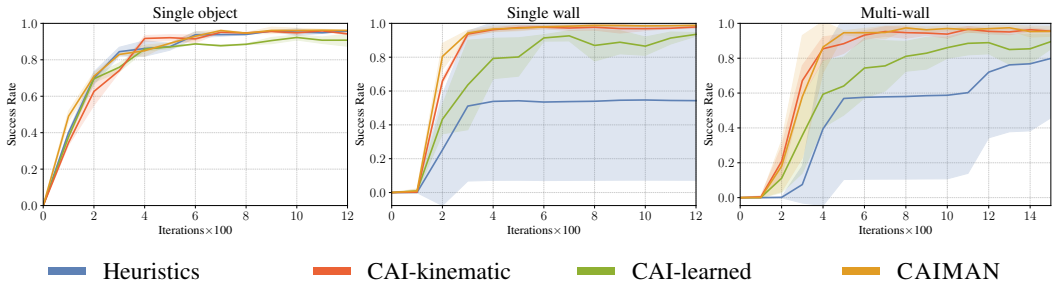


Figure F.1: Policy success rate during training under a dense task reward for all methods. All experiments utilize a minimum of 2 seeds.

G Additional Experiment: Cart

To demonstrate CAIMAN's effectiveness in handling complex objects with asymmetric dynamics, we provide an additional experiment: pushing a simplified cart with two normal and two caster wheels, as shown in Fig. G.1. The scene configuration is identical to that of the Single object task, with the box being replaced by the cart. Our results demonstrate that CAIMAN is more sample efficient and achieves a higher asymptotic performance (approx. 90% success rate) than most baselines and is on par with the best competitor. Notably, this experiment did not require any changes to the framework or physics prior, highlighting the capability of the learned model in estimating complex dynamics and the effectiveness of CAIMAN for learning with irregular objects. This environment has no obstacles, but we hypothesize that cluttered settings with walls would further emphasize CAIMAN's advantages over baselines.

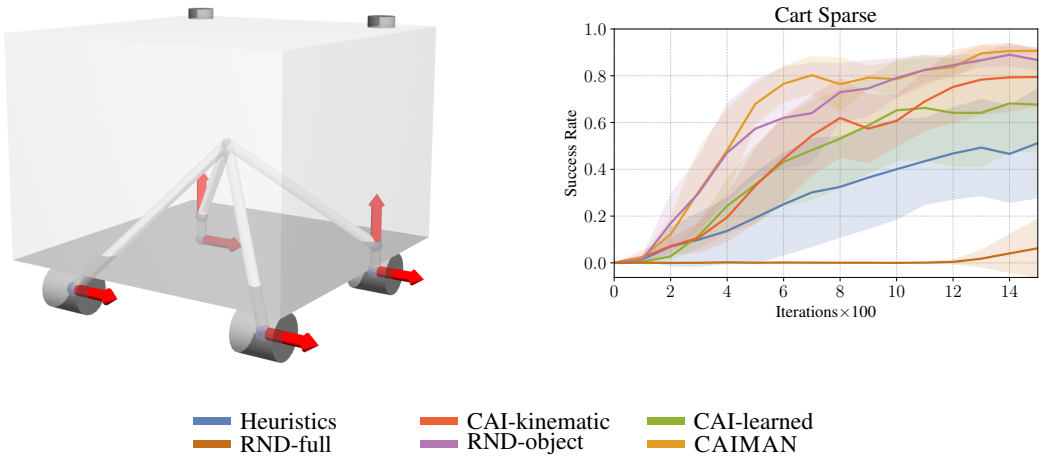


Figure G.1: *Left*: A simplified cart object. *Right*: Single cart pushing results.