SigLoMa: Learning Open-World Quadrupedal Loco-Manipulation from Ego-Centric Vision

Shiyi Chen[1], **Haiyi Liu**[1], **Mingye Yang**[2], **Jiaqi Zhang**[1], **Debing Zhang**[1,*]
[1]Tsinghua University [2]Imperial College London
https://11chens.github.io/SigLoMa/

Abstract: Designing an open-world quadrupedal loco-manipulation system is highly challenging. Traditional reinforcement learning frameworks utilizing exteroception often suffer from extreme sample inefficiency and massive sim-to-real gaps. Furthermore, the inherent latency of visual tracking fundamentally conflicts with the high-frequency demands of precise floating-base control. Consequently, existing systems lean heavily on expensive external motion capture and off-board computation. To eliminate these dependencies, we present SigLoMa, a fully onboard, ego-centric vision-based pick-and-place framework. At the core of SigLoMa is the introduction of Sigma Points, a lightweight geometric representation for exteroception that guarantees high scalability and native sim-to-real alignment. To bridge the frequency divide between slow perception and fast control, we design an ego-centric Kalman Filter to provide robust, high-rate state estimation. On the learning front, we alleviate sample inefficiency via an Active Sampling Curriculum guided by Hint Poses, and tackle the robot's structural visual blind spots using temporal encoding coupled with simulated random-walk drift. Real-world experiments validate that, relying solely on a 5Hz (200 ms latency) open-vocabulary detector, SigLoMa successfully executes dynamic loco-manipulation across multiple tasks, achieving performance comparable to expert human teleoperation.

Keywords: Quadrupedal Loco-manipulation, Ego-centric Vision, Sim-to-Real

1 Introduction

Equipping quadrupedal robots with manipulation capabilities holds immense practical value for unstructured environments. However, precise loco-manipulation coupled with a floating base remains fundamentally challenging. While recent works have explored Reinforcement Learning (RL) methods in simulation to coordinate locomotion [1, 2, 3, 4, 5] and manipulation [6, 7, 8], external perception remains a prominent bottleneck. In simulation, rendering dense visual inputs incurs high memory overhead, drastically reduces environment parallelism, and leads to highly inefficient training. Furthermore, deploying such vision-based policies introduces a profound sim-to-real gap [9, 10].

Most current perception pipelines rely on dense geometric information (e.g., depth maps [11, 12] or point clouds [13, 14]) to narrow this gap; however, they inherently discard semantic understanding, severely limiting practical autonomy. Alternative modular pipelines incorporate visual object detectors, but running open-vocabulary trackers on constrained onboard computers yields control frequencies too low for dynamic manipulation. To guarantee high-frequency state estimation, mainstream solutions often rely on high-performance off-board compute nodes or external motion capture systems [15, 16]. While indispensable for extreme high-speed tasks (e.g., robotic sports), such infrastructure-heavy setups restrict the autonomy of legged systems in everyday spaces. Furthermore, while applying filtering algorithms is a standard remedy for low-frequency signals, utilizing them directly on a walking quadruped is notoriously difficult; traditional filters struggle to decouple the target's relative movement from the severe ego-motion of the floating base, leading to rapid

*Corresponding author

drift. Consequently, many recent frameworks bypass the continuous floating-base control challenge by stopping the base first and relying primarily on quasi-static robotic arm operations [17, 8, 18].

To address these interconnected challenges, we propose SigLoMa, a fully onboard quadrupedal loco-manipulation framework tailored for unstructured, open-world scenarios. SigLoMa is built upon three core pillars. First, we introduce Sigma Points, a lightweight, category-agnostic geometric representation that bridges upstream open-vocabulary semantic detectors and downstream continuous control, enabling efficient simulation training without dense visual rendering. Second, we design an ego-centric Kalman Filter (KF) that bridges the frequency gap between low-rate embodied perception outputs and the high-frequency demands of continuous floating-base control. By explicitly compensating for camera ego-motion, the KF separates target motion from base motion and produces robust, high-rate state estimates. Finally, we train a robust RL policy with an Active Sampling Curriculum (ASC) and Hint Poses to learn object-specific approach trajectories without heavy reward engineering. To improve reliability during the final grasping phase, SigLoMa incorporates a memory-augmented formulation that mitigates structural blind spots and sim-to-real open-loop drift.

Our core contributions are summarized as follows:

1. **A Hardware-Efficient Loco-Manipulation System**: SigLoMa is a fully onboard vision-based framework that enables precise floating-base control without external motion capture or off-board computation;
2. **An Ego-Centric Geometric Perception Pipeline**: a state estimation architecture built on Sigma Points and an ego-centric KF, which bridges embodied semantic tracking and fast control through explicit ego-motion compensation;
3. **A Robust Long-Horizon Learning Curriculum**: an integrated RL strategy that uses an ASC and hint poses, together with a temporal memory network, to overcome sparse rewards and improve robustness under structural visual occlusions.

2 Related Work

Perception-Aware Legged Control. Integrating perception into reinforcement learning has driven breakthroughs in legged locomotion. To mitigate the RGB sim-to-real gap, early works leveraged elevation or depth maps via privileged two-stage distillation (e.g., DAgger [19]) [20, 21, 2, 3, 11, 22, 23, 24, 25, 26], occasionally incorporating LiDAR for enhanced foothold planning [14, 27, 28].

Transitioning to loco-manipulation, however, necessitates both geometric and semantic understanding. Many systems address this by training with privileged simulated object states [7, 29, 30, 31, 32, 33] and deploying task-specific detectors. Most notably, robotic badminton [7] utilizes an Extended Kalman Filter (EKF) for shuttlecock trajectory estimation, but heavily relies on Multi-Sensor Fusion and CompSLAM to maintain a rigidly stable global coordinate system. While effective for structured sports, this strict reliance on drift-free global estimation increases deployment complexity and restricts open-world mobility. Furthermore, such stringent tracking precludes generic, open-vocabulary vision. Consequently, achieving long-horizon floating-base manipulation using exclusively low-frequency semantic detectors on constrained onboard compute remains unresolved.

Quadruped Loco-Manipulation Pipelines. Current loco-manipulation architectures generally fall into implicit learning frameworks and modular decoupled designs. Implicit approaches, such as end-to-end RL [8] and behavior-cloning via Action Chunking Transformers [18], directly map observations to actions. However, they suffer from sample inefficiency, heavy reliance on teleoperation, and their latent representations preclude integration with explicit high-frequency state estimators.

Modular systems typically outfit the quadruped with a robotic arm [17, 6, 34, 35, 36], executing a "Maps-then-reach" strategy. While transferring the accuracy burden to a dedicated manipulator relaxes base tracking requirements, it introduces substantial hardware cost, payload constraints, and power overhead. A highly cost-effective alternative mounts a gripper directly onto the base, which

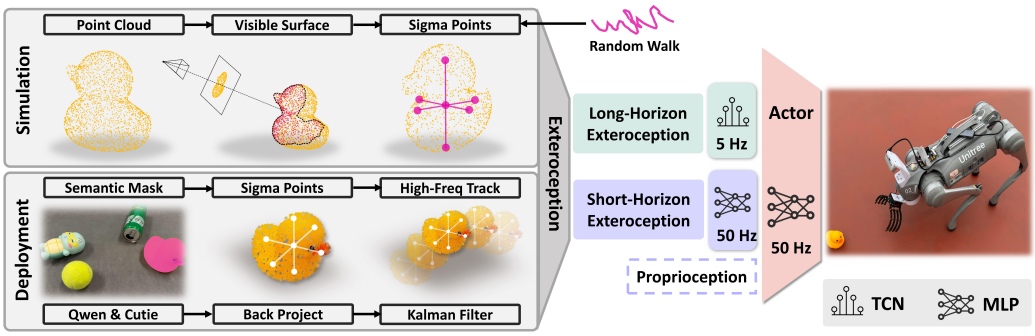

Figure 1: **Overall System Architecture. (1) Perception Design:** In simulation, 50 Hz Sigma Points are extracted from object point clouds via visible surface computation and PCA, with random-walk noise injected in blind spots to ensure robust real-world open-loop predictions. In deployment, semantic masks from visual tracking are back-projected to the camera frame to generate 5 Hz Sigma Points, which are then upsampled and latency-compensated to 50 Hz by an ego-centric Kalman Filter. **(2) Network Architecture:** Dual encoders process the Sigma Points across different frequencies and time horizons. Finally, an Actor network translates these temporal features into continuous locomotion and posture commands.

mandates continuous, highly precise base regulation. Existing systems utilizing this morphology either restrict manipulation to task-specific dynamic interceptions (e.g., catching spheres) [29] or heavily rely on global external cameras and low-frequency perception [15].

In contrast, SigLoMa maximizes this arm-free morphology using exclusively onboard, ego-centric perception. By providing the high-frequency, low-latency state estimation essential for stringent continuous base control, it circumvents both the hardware bloat of armed quadrupeds and the external infrastructure reliance of prior base-mounted pipelines.

3 Method

Designing an open-world quadrupedal pick-and-place solution presents dual overarching challenges. For simulation training, the RL policy must generalize across diverse object geometries to interface seamlessly with high-level Vision Language Models (VLMs). For real-world deployment, maintaining high manipulation precision on a highly dynamic floating base imposes strict latency and frequency requirements; even minor visual tracking delays translate into centimeter-level base deviation errors, frequently causing grasp failures. To address these issues, we propose SigLoMa, a hierarchical and highly adaptive framework: (1) prompt-driven VLM parsing for open-vocabulary target detection; (2) an ego-centric KF handling camera motion compensation, frequency amplification, and delay masking; and (3) a robust high-level RL policy utilizing computationally lightweight Sigma Points to formulate dynamic velocity and posture commands for reaching the optimal operational pose.

3.1 Hierarchical System Overview

As depicted in Fig. 1, the SigLoMa pipeline initiates with a VLM processing raw user language commands to instantiate a `pick_target` and a `place_target`. We then feed these zero-shot anchors into an onboard open-vocabulary tracking model, which yields low-frequency semantic masks. Because raw visual masks are computationally prohibitive for large-scale simulation and highly susceptible to sim-to-real domain shifts, we distill this raw perception into a sparse set of geometric feature points, termed *Sigma Points*. Operating on these sparse features, a Kalman Filter (KF) bridges the low-frequency visual updates to output high-frequency, low-latency target state estimates. Finally, these estimates are routed to the RL policy network to generate continuous velocity and posture commands, achieving precise grasping and placing maneuvers.

3.2 Exteroceptive State Representation via Sigma Points

To capture object geometry while maintaining computational efficiency, we represent target objects via Sigma Points—a sparse, mathematically derived set of representative features that effectively encapsulate bounding volume and spatial orientation.

Let $\mathcal{P} = \{{}^{C}\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$ be the dense object point cloud transformed into the camera coordinate frame $\{C\}$, with corresponding surface normals ${}^{C}\mathbf{n}_i$. We first extract the visible point set $\mathcal{V}$ based on strict geometric back-face culling and Field-of-View (FoV) constraints. The visibility condition dictates that the surface normal must oppose the viewing direction ${}^{C}\mathbf{p}_i$:

$$\mathcal{V} = \left\{ {}^{C}\mathbf{p}_i \in \mathcal{P} \mid {}^{C}\mathbf{n}_i \cdot {}^{C}\mathbf{p}_i < 0 \wedge \text{FoV}({}^{C}\mathbf{p}_i) = \text{True} \right\}$$

In simulation, points uniformly cover 3D surfaces, whereas real-world points back-projected from 2D masks suffer from perspective-induced downsampling on slanted and distant faces. To bridge this sim-to-real density gap, we assign a geometry-aware weight w_i to each visible point. This weight emulates the real camera pixel distribution by accounting for the differential solid angle subtended by surface patches [37]: $w_i = \max\left(0, \frac{-{}^{C}\mathbf{n}_i \cdot {}^{C}\mathbf{p}_i}{\|{}^{C}\mathbf{p}_i\|^3}\right)$. By incorporating w_i into a weighted Principal Component Analysis (PCA) on $\mathcal{V}$ [38], the extracted Sigma Points in simulation achieve a statistical distribution tightly aligned with real-world pixel-level observations. This alignment ensures the downstream RL policy remains invariant to the underlying point sampling source. The weighted centroid ${}^{C}\boldsymbol{\mu}$ and the spatial covariance matrix $\boldsymbol{\Sigma}_p$ are computed as:

$${}^{C}\boldsymbol{\mu} = \frac{\sum_{i\in\mathcal{V}} w_i {}^{C}\mathbf{p}_i}{\sum_{i\in\mathcal{V}} w_i}, \quad \boldsymbol{\Sigma}_p = \frac{\sum_{i\in\mathcal{V}} w_i ({}^{C}\mathbf{p}_i - {}^{C}\boldsymbol{\mu})({}^{C}\mathbf{p}_i - {}^{C}\boldsymbol{\mu})^T}{\sum_{i\in\mathcal{V}} w_i}$$

Through eigendecomposition $\boldsymbol{\Sigma}_p\mathbf{e}_k = \lambda_k\mathbf{e}_k$, we extract the eigenvalues λ_k and eigenvectors $\mathbf{e}_k$ ($k \in \{1, 2, 3\}$). We systematically sample two points per principal axis alongside the geometric centroid, yielding a geometric feature set $\mathbf{S} = \{\mathbf{s}_j\}_{j=0}^{6}$:

$${}^{C}\mathbf{s}_0 = {}^{C}\boldsymbol{\mu}, \quad {}^{C}\mathbf{s}_{k,\pm} = {}^{C}\boldsymbol{\mu} \pm \alpha\sqrt{\lambda_k}\mathbf{e}_k$$

where α is a scaling factor. To maintain temporal consistency with downstream filtering and control modules, we denote this observed point set at any given camera frame t as ${}^{C_t}\mathbf{S}_t^{\text{obs}}$. During RL training, we analytically compute pseudo-visible surfaces relative to the camera frame. In the real world, semantic masks are back-projected into the 3D camera coordinate frame and undergo the identical PCA process to obtain the actual Sigma Points. The processing pipeline is illustrated in Fig. 1.

3.3 Ego-Centric State Estimation via Kalman Filtering

Significant visual tracking latency severely restricts dynamic manipulation. Due to the large-range motion of the floating base, traditional methods often resort to external motion capture systems or only predict the trajectory of moving objects. To this end, we implement an ego-centric KF updating locally within the camera frame, utilizing Visual Odometry (VO) to mitigate base perturbation.

To track the target's geometric structure, we apply the filtering process uniformly to all elements of the Sigma Points set $\mathbf{S}$. For each Sigma Point $\mathbf{s}_j$ ($j \in \{0, \ldots, 6\}$), the system state is defined directly in the current camera frame $\{C_t\}$ as ${}^{C_t}\mathbf{x}_{j,t} = [({}^{C_t}\mathbf{s}_{j,t})^T, ({}^{C_t}\mathbf{v}_{j,t})^T]^T \in \mathbb{R}^6$, where ${}^{C_t}\mathbf{s}_{j,t}$ and ${}^{C_t}\mathbf{v}_{j,t}$ represent its 3D position and relative velocity, respectively. The core innovation of our filter lies in decoupling the state transition into two distinct physical processes: point motion prediction and camera ego-motion compensation.

For the prediction step, we adopt a linear kinematic approximation to predict the j-th point's short-term displacement relative to the previous frame $\{C_{t-1}\}$:

$${}^{C_{t-1}}\mathbf{s}_{j,t}^{-} = {}^{C_{t-1}}\mathbf{s}_{j,t-1} + {}^{C_{t-1}}\mathbf{v}_{j,t-1}\Delta t \tag{1}$$

Simultaneously, we rectify the shifts in the sensory frame caused by the robot's base movement. Given the sequential world-to-camera poses ${}^{W}\mathbf{T}_{C_{t-1}}$ and ${}^{W}\mathbf{T}_{C_t}$ provided by the VO, we extract the

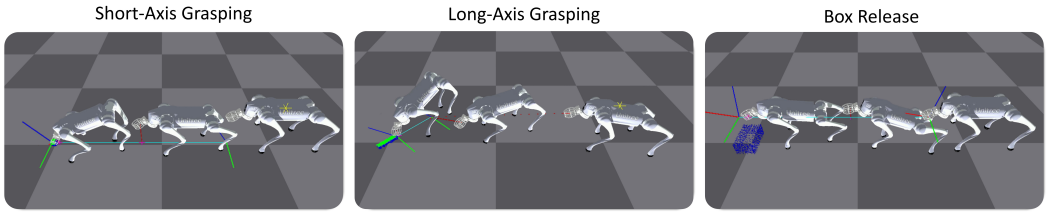

Figure 2: **Task Taxonomy and Pose Design.** Optimal terminal poses are located on the objects, while hint poses are positioned mid-way. The green lines connecting them indicate the suggested motion trajectories. Object point clouds are visualized as blue points, with real-time computed visible surfaces highlighted in green. Notably, for long-axis objects, the hint pose implicitly guides the robot to maneuver from the front to the side to approach the short-axis grasping point.

relative transformation ${}^{C_t}\mathbf{T}_{C_{t-1}} = ({}^{W}\mathbf{T}_{C_t})^{-1}{}^{W}\mathbf{T}_{C_{t-1}}$. Using the rotational component ${}^{C_t}\mathbf{R}_{C_{t-1}}$ and translational component ${}^{C_t}\mathbf{t}_{C_{t-1}}$ of this relative transformation, we map each predicted point state directly into the new camera perspective:

$$ {}^{C_t}\hat{\mathbf{s}}_{j,t} = {}^{C_t}\mathbf{R}_{C_{t-1}}\,{}^{C_{t-1}}\mathbf{s}^{-}_{j,t} + {}^{C_t}\mathbf{t}_{C_{t-1}}, \quad {}^{C_t}\hat{\mathbf{v}}_{j,t} = {}^{C_t}\mathbf{R}_{C_{t-1}}\,{}^{C_{t-1}}\mathbf{v}_{j,t-1} \tag{2} $$

When the visual observation arrives, the newly computed Sigma Points from the PCA serve as measurement updates to correct their respective prediction errors. This filtering process yields the robust Sigma Points set ${}^{C_t}\mathbf{S}_t^{\text{obs}}$ required by the downstream RL policy. It abolishes the dependency on a tightly unified global coordinate system and bounds the residual error to the inter-frame increment, resulting in low-delay geometric state estimations. Detailed mathematical derivations of the entire filtering process are provided in Appendix B.

3.4 Policy Formulation and Robust Learning Framework

Task Taxonomy and MDP Setup. We formulate the loco-manipulation problem across a discrete task space $\mathcal{T} = \{\tau_{\text{long}}, \tau_{\text{short}}, \tau_{\text{release}}\}$, which is visually summarized in Fig. 2. Specifically, we deploy dozens of daily items and regular objects to evaluate the system. For picking operations, an object whose longest axis exceeds the maximum gripper aperture is classified as a long-axis target requiring τ_{long}, which dictates an orthogonal approach trajectory; conversely, smaller targets default to τ_{short}. For placing operations, the system universally executes the object release task τ_{release}.

To solve these tasks, our high-level RL policy maps observations to actions at each control step t. As illustrated in Fig. 1, the observation space is decoupled into proprioception and exteroception. The proprioceptive state $\mathbf{o}_t^{\text{prop}} = [{}^{B}\mathbf{g}_t, {}^{B}\mathbf{v}_t, {}^{B}\boldsymbol{\omega}_t, \mathbf{a}_{t-1}, c_{\text{task}}]$ encapsulates the robot's base-projected gravity, linear and angular velocities, the previous action, and the task phase flag. The exteroceptive state relies entirely on the tracking of Sigma Points ${}^{C_t}\mathbf{S}_t^{\text{obs}} \in \mathbb{R}^{7\times 3}$, which are processed via a dual-horizon architecture. Specifically, a short-horizon encoder processes a high-frequency history of H_{short} frames to overcome perception noise, while a long-horizon encoder processes a low-frequency history of H_{long} frames to provide a memory window of effective perception within blind spots. These extracted temporal features are fused with the proprioceptive state and passed to the Actor network, which outputs continuous base velocities and body pitch commands $\mathbf{a}_t = [v_x, v_y, \omega_z, \theta_y]^T$. Finally, a pre-trained low-level controller [39] executes these commands. We train the policy using task constraints and stability regularizations to achieve optimal terminal alignment and stable locomotion, detailing formulations in Appendix D.

Trajectory Shaping and Active Sampling. To combat the inherent sample inefficiency and sparse rewards of long-horizon tasks, we introduce two critical training optimizations. First, to regularize the policy's approach behaviors, we establish an optimal terminal pose for open-loop grasp triggering for each $\tau \in \mathcal{T}$. Since standard sparse rewards poorly constrain the strict approach trajectories dictated by object geometry (e.g., elongated objects mandate orthogonal approaches due to limited gripper aperture), we introduce *hint poses* to implicitly regularize trajectory rollouts, as

shown in Fig. 2. Serving as intermediate orientational waypoints, these hint poses enforce target pre-alignment, naturally shaping the approach to ensure smooth convergence to the optimal pose.

Second, we utilize an Active Sampling Curriculum (ASC) that dynamically shifts the training focus from guided skill-bootstrapping to aggressive hard-negative mining. Inspired by biological learning, we design a *near-optimal* initialization that functions as an active "feeding" mechanism. By intentionally spawning objects in close proximity to the optimal pose during early training, it provides immediate, dense rewards that drastically lower the initial exploration barrier, conceptually akin to parental nurturing. As the policy matures, this feeding mechanism naturally decays, seamlessly shifting the curriculum's focus toward a *failure-replay* distribution to aggressively eliminate bottleneck behaviors.

This curriculum transition is mathematically driven by a normalized task competency metric $\rho = \min(s/s_{\text{thresh}}, 1.0)$. Here, s denotes the running success rate, evaluated via dual geometric boundaries: a rollout succeeds if the terminal pose deviation falls below the success boundary and is registered as a failure when it violates the failure boundary at timeout. The formal definitions of these boundaries are provided in Appendix C. s_{thresh} is a predefined curriculum threshold. The sampling probability P_i for initialization type $i \in \{\text{near-optimal}, \text{failure-replay}\}$ transitions exponentially from $p_{i,\text{start}}$ to $p_{i,\text{end}}$:

$$P_i(\rho) = p_{i,\text{end}} + (p_{i,\text{start}} - p_{i,\text{end}}) \exp(-\lambda_{\text{ASC}}\rho)$$

where λ_{ASC} governs the transition rate. Consequently, the feeding probability exponentially decays ($p_{i,\text{start}} > p_{i,\text{end}}$) while the failure-replay probability is exponentially prioritized ($p_{i,\text{start}} < p_{i,\text{end}}$).

Memory-Augmented Blind-Spot Handling. Terminal grasping inevitably pushes the target out of the camera's FoV. To robustly simulate the ensuing open-loop KF drift during this out-of-FoV phase, we inject a cumulative, clamped random-walk noise $\mathbf{d}_t \in \mathbb{R}^3$ uniformly into the true Sigma Points ${}^{C_t}\mathbf{S}_t^{\text{true}}$:

$$\mathbf{d}_t = \text{clip}\big(\mathbf{d}_{t-1} + \mathcal{N}(\mathbf{0}, \sigma_{\text{drift}}^2\mathbf{I}), -d_{\max}, d_{\max}\big), \quad {}^{C_t}\mathbf{S}_t^{\text{obs}} = {}^{C_t}\mathbf{S}_t^{\text{true}} + \mathbf{d}_t \tag{3}$$

where σ_{drift} is the standard deviation governing the random-walk variance, $d_{\max}$ restricts the maximum positional drift, and $\mathbf{d}_t$ resets to zero upon visual tracking recovery. To maintain high-precision manipulation through this blind spot, a sufficient temporal receptive field is essential. To circumvent the prohibitive overhead of buffering at the 50Hz control rate, we maintain a lightweight 2s perception memory down-sampled to 5Hz. Processed via a Temporal Convolutional Network (TCN), this historical buffer effectively encapsulates spatiotemporal context, natively bridging the sim-to-real blind-spot gap under strict onboard compute limits.

4 Experimental Results

4.1 Simulation Setup and Ablation Studies

The RL policy is trained using Proximal Policy Optimization (PPO) [40] in Isaac Gym [41]. To ensure robust open-world generalization, the training environment incorporates a diverse object distribution, encompassing geometric primitives (e.g., cuboids, spheres, boxes, and cylinder baskets) and 27 everyday household items from the YCB dataset [42]. Furthermore, the initial 6-DoF poses of both the robot and the target objects are uniformly randomized at the onset of each episode. Despite this extensive morphological and spatial randomization, SigLoMa's compact Sigma Point representation and the ASC enable exceptional sample efficiency; the policy fully converges in approximately 6 hours on a single NVIDIA RTX 4090 GPU.

We evaluate the impact of SigLoMa's core algorithmic components through extensive ablation studies. Specifically, we investigate variants operating without hint poses (**No Hint**), without the ASC (**No ASC**), and with modified blind-spot memory architectures (e.g., completely removing the TCN encoder (**No TCN**), or substituting the TCN with an MLP (**Enc-MLP**) or a GRU (**Enc-GRU**)). These configurations were evaluated across a diverse set of objects, mapped to task space $\mathcal{T}$.

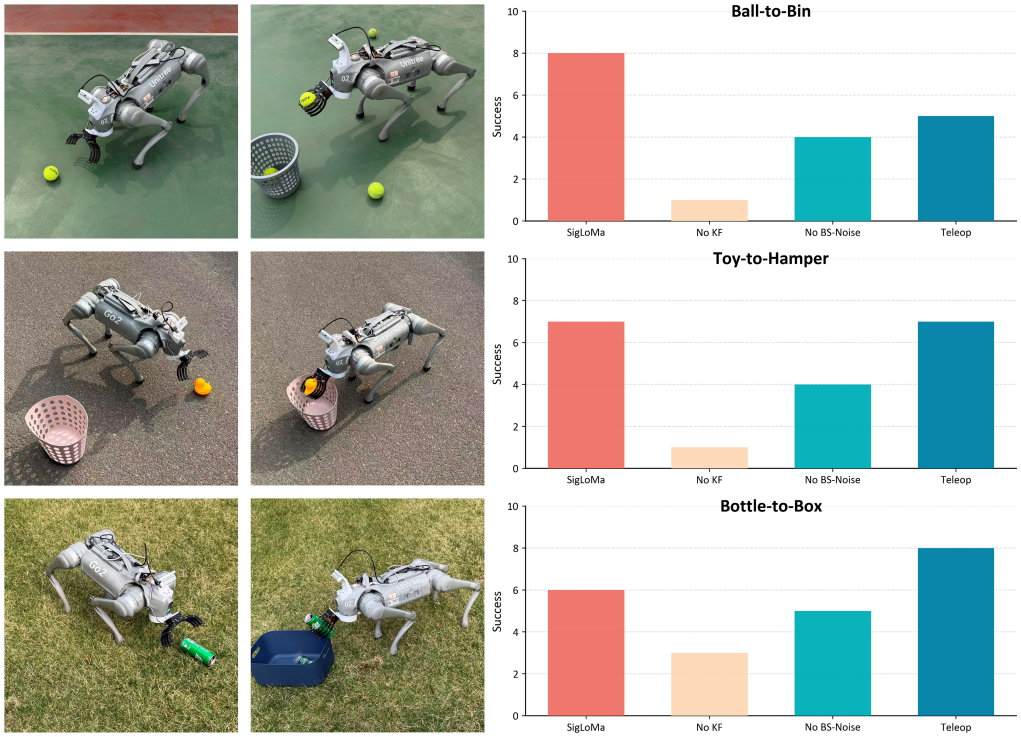

Figure 3: **Real-World Experimental Results.** Snapshots of the pick and place phases across three tasks, alongside their end-to-end success counts.

Table 1: Simulation Ablation Studies. Success is defined as the robot successfully stopping at the task-specific optimal pose (i.e., multi-dimensional tracking error strictly less than predefined success thresholds). We report the mean success rate (%) and standard deviation for each condition. Each experiment is evaluated over 100 independent trials, and the reported values are averaged across 3 independent evaluation runs with randomized initial poses for both the robot and the object.

	Short-Axis Grasp			Long-Axis Grasp			Target Release	
Method	Cuboid	Apple	Sugar	Sphere	Conditioner	Shampoo	Basket	Box
SigLoMa	$96.3_{\pm 1.5}$	$\mathbf{97.0_{\pm 1.0}}$	$\mathbf{96.0_{\pm 1.0}}$	$\mathbf{87.7_{\pm 3.2}}$	$83.3_{\pm 2.5}$	$\mathbf{85.0_{\pm 2.0}}$	$\mathbf{95.3_{\pm 1.5}}$	$\mathbf{92.7_{\pm 2.1}}$
Enc-MLP	$\mathbf{96.7_{\pm 1.5}}$	$96.0_{\pm 2.0}$	$95.3_{\pm 2.1}$	$83.3_{\pm 2.1}$	$79.3_{\pm 2.5}$	$82.3_{\pm 3.5}$	$91.0_{\pm 1.7}$	$87.3_{\pm 2.5}$
Enc-GRU	$82.7_{\pm 3.2}$	$81.3_{\pm 3.5}$	$83.3_{\pm 2.1}$	$72.0_{\pm 3.6}$	$67.7_{\pm 4.2}$	$69.3_{\pm 3.1}$	$82.3_{\pm 3.1}$	$78.3_{\pm 3.5}$
No TCN	$89.3_{\pm 2.1}$	$88.0_{\pm 2.6}$	$90.0_{\pm 2.0}$	$82.0_{\pm 2.6}$	$\mathbf{86.3_{\pm 2.1}}$	$83.3_{\pm 2.5}$	$88.3_{\pm 2.1}$	$84.3_{\pm 2.1}$
No ASC	$78.0_{\pm 2.0}$	$76.3_{\pm 2.5}$	$77.7_{\pm 2.1}$	$44.0_{\pm 4.0}$	$39.3_{\pm 4.5}$	$43.3_{\pm 4.5}$	$77.0_{\pm 3.0}$	$73.0_{\pm 3.6}$
No Hint	$72.7_{\pm 2.5}$	$71.3_{\pm 3.1}$	$73.0_{\pm 2.6}$	$16.0_{\pm 3.6}$	$12.0_{\pm 2.0}$	$14.3_{\pm 2.5}$	$72.3_{\pm 2.5}$	$68.3_{\pm 2.5}$

Results indicate that the introduction of hint poses and the ASC serves as the primary catalyst for learning long-horizon, long-axis grasping maneuvers. Without these mechanisms, the policy consistently fails to explore successful perpendicular approach trajectories. Furthermore, ablating the temporal memory window or substituting the sequential encoder with an MLP demonstrably degrades grasping success. This indicates that retaining representations of previously valid visual observations is crucial for enabling the robot to maintain a stable posture.

Moreover, the GRU variant underperforms the TCN. We attribute this to the GRU's recursive memory, which is highly sensitive to the non-stationary data distributions in RL. In contrast, the TCNs's feedforward structure, operating over a fixed receptive field without accumulating hidden state errors, inherently provides a more robust temporal representation. Interestingly, during testing, we

observed emergent active-perception behaviors: the agent inherently maintains a "head-up" pitch at a distance to maximize FoV retention, performs granular lateral corrections upon proximity, and executes a smooth deceleration into the blind spot to minimize internal KF drift.

4.2 Real-World Hardware Deployment

We deployed SigLoMa on a Unitree Go2 quadruped, equipped with a RealSense D435i camera, a custom low-cost gripper, and an onboard NVIDIA Jetson Orin NX. VO is provided by Isaac ROS Visual SLAM at 60 Hz. For the perception stack, we leverage the Qwen 3.5 VLM API [43] to extract initial semantic masks via zero-shot reasoning. These masks initialize Cutie [44], which runs entirely onboard to provide continuous target tracking at a low frequency of 5 Hz. Detailed hardware configurations and task execution workflows are deferred to the Appendix A.

As illustrated in Fig. 3, we documented performance across three distinct pick-and-place tasks representing the SigLoMa taxonomy $\mathcal{T}$: *ball-to-bin* ($\tau_{\text{short}} \rightarrow \tau_{\text{release}}$), *toy-to-hamper* ($\tau_{\text{short}} \rightarrow \tau_{\text{release}}$), and *water bottle-to-box* ($\tau_{\text{long}} \rightarrow \tau_{\text{release}}$). Each setting underwent 10 trials, from which we recorded the first-attempt end-to-end success counts. To evaluate the effectiveness of our proposed methods for closing the sim-to-real gap, we conducted ablation studies isolating the KF (**No KF**) and the blind-spot random-walk noise (**No BS-Noise**), ultimately comparing the SigLoMa autonomous pipeline against human teleoperation (**Teleop**).

The real-world executions definitively highlight the indispensability of SigLoMa's ego-centric state estimation framework. The KF effectively compensates for processing latency and bridges the low-frequency vision backbone, ensuring real-time locomotion reactivity. Furthermore, injecting random-walk noise during simulation training proves crucial for modeling the KF's open-loop prediction drift. This targeted augmentation explicitly reduces the sim-to-real gap, enabling robust policy execution even within visual blind spots. Ultimately, the fully onboard SigLoMa autonomous pipeline achieves highly precise floating-base maneuvers, yielding end-to-end success counts that perform on par with human teleoperation.

5 Conclusion

In this paper, we present SigLoMa, a low-cost, open-world solution for quadrupedal loco-manipulation that eliminates the need for external tracking and high-compute platforms. To achieve this, we introduce the Sigma Point representation for efficient RL and sim-to-real transfer, alongside an ego-centric Kalman Filter that bridges the frequency gap between semantic detection and continuous control. On the learning front, an ASC guided by hint poses and a TCN with random-walk noise accelerate multi-task convergence and robustly handle visual blind spots. Real-world experiments demonstrate that SigLoMa autonomously adapts grasping strategies to diverse objects, performing dynamic pick-and-place with success rates comparable to expert human teleoperation.

5.1 Limitations and Future Work

Despite strong sim-to-real transfer, SigLoMa exhibits a few notable limitations. First, manipulation is currently constrained to flat surfaces; future work will focus on integrating local elevation maps and training the policy from scratch to maintain the stability of the gripper's pose on uneven terrains. Second, acting primarily as a local visual servoing policy, the framework lacks long-horizon navigation capabilities, which could be addressed by incorporating a semantic-based navigation module to enable long-distance traversal and adaptive obstacle avoidance. Third, as a cascaded multi-module system, the overall grasping precision is susceptible to perception instability. Specifically, errors from frequency fluctuations or dropped frames in any single module can propagate downstream. To mitigate this, future efforts could explore optimizing system-level integration and enhancing the temporal memory architecture to build resilience against such sensory data fluctuations.

References

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

[2] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao. Robot parkour learning. In *Conference on Robot Learning (CoRL)*, 2023.

[3] X. Cheng, K. Shi, A. Agarwal, and D. Pathak. Extreme parkour with legged robots. *arXiv preprint arXiv:2309.14341*, 2023.

[4] J. Long, W. Yu, Q. Li, Z. Wang, D. Lin, and J. Pang. Learning h-infinity locomotion control. *arXiv preprint arXiv:2404.14405*, 2024.

[5] R. Huang, S. Zhu, Y. Du, and H. Zhao. Moe-loco: Mixture of experts for multitask locomotion, 2025. URL https://arxiv.org/abs/2503.08564.

[6] 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.

[7] Y. Ma, A. Cramariuc, F. Farshidian, and M. Hutter. Learning coordinated badminton skills for legged manipulators. *Science Robotics*, 10(102), May 2025. ISSN 2470-9476. doi:10.1126/scirobotics.adu3922. URL http://dx.doi.org/10.1126/scirobotics.adu3922.

[8] M. Liu, Z. Chen, X. Cheng, Y. Ji, R.-Z. Qiu, R. Yang, and X. Wang. Visual whole-body control for legged loco-manipulation, 2024. URL https://arxiv.org/abs/2403.16967.

[9] W. Yu, D. Jain, A. Escontrela, A. Iscen, P. Xu, E. Coumans, S. Ha, J. Tan, and T. Zhang. Visual-locomotion: Learning to walk on complex terrains with vision. In *Conference on Robot Learning*, pages 1691–1702. PMLR, 2022.

[10] A. Agarwal, A. Kumar, J. Malik, and D. Pathak. Legged locomotion in challenging terrains using egocentric vision. In *Conference on Robot Learning*, pages 403–415. PMLR, 2023.

[11] 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.

[12] ETH-PBL. Robust reinforcement learning-based locomotion for resource-constrained quadrupeds with exteroceptive sensing. *arXiv preprint arXiv:2505.12537*, 2025.

[13] Q. Yuan, Z. Cao, M. Cao, and K. Li. Reasan: Learning reactive safe navigation for legged robots, 2025. URL https://arxiv.org/abs/2512.09537.

[14] Z. Wang, T. Ma, Y. Jia, X. Yang, J. Zhou, W. Ouyang, Q. Zhang, and J. Liang. Omni-perception: Omnidirectional collision avoidance for legged locomotion in dynamic environments, 2025. URL https://arxiv.org/abs/2505.19214.

[15] Q. Wu, Z. Fu, X. Cheng, X. Wang, and C. Finn. Helpful doggybot: Open-world object fetching using legged robots and vision-language models, 2024. URL https://arxiv.org/abs/2410.00231.

[16] C. Liu, L. Jiang, Y. Wang, K. Yao, J. Fu, and X. Ren. Humanoid whole-body badminton via multi-stage reinforcement learning, 2025. URL https://arxiv.org/abs/2511.11218.

[17] J. Wang, J. Rajabov, C. Xu, Y. Zheng, and H. Wang. Quadwbg: Generalizable quadrupedal whole-body grasping, 2025. URL https://arxiv.org/abs/2411.06782.

[18] R.-Z. Qiu, Y. Song, X. Peng, S. A. Suryadevara, G. Yang, M. Liu, M. Ji, C. Jia, R. Yang, X. Zou, and X. Wang. Wildlma: Long horizon loco-manipulation in the wild, 2025. URL https://arxiv.org/abs/2411.15131.

[19] S. Ross, G. J. Gordon, and J. A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

[20] A. Loquercio, A. Kumar, and J. Malik. Learning visual locomotion with cross-modal supervision. *arXiv preprint arXiv:2211.03785*, 2022.

[21] D. Hoeller, N. Rudin, C. Choy, A. Anandkumar, and M. Hutter. Neural scene representation for locomotion on structured terrain, 2022. URL https://arxiv.org/abs/2206.08077.

[22] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis. Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control. *IEEE Transactions on Robotics*, 38(5):2908–2927, 2022.

[23] H. Duan, B. Pandit, M. S. Gadde, B. J. Van Marum, J. Dao, C. Kim, and A. Fern. Learning vision-based bipedal locomotion for challenging terrain. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 56–62. IEEE, 2024.

[24] F. Chen, R. Wan, P. Liu, N. Zheng, and B. Zhou. Vmts: Vision-assisted teacher-student reinforcement learning for multi-terrain locomotion in bipedal robots. *arXiv preprint arXiv:2503.07049*, 2025.

[25] D. Wang, X. Wang, X. Liu, J. Shi, Y. Zhao, C. Bai, and X. Li. More: Mixture of residual experts for humanoid lifelike gaits learning on complex terrains. *arXiv preprint arXiv:2506.08840*, 2025.

[26] R. Fawcett et al. Vital: Vision-based terrain-aware locomotion for legged robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[27] J. Long, J. Ren, M. Shi, Z. Wang, T. Huang, P. Luo, and J. Pang. Learning humanoid locomotion with perceptive internal model, 2024. URL https://arxiv.org/abs/2411.14386.

[28] Y. Jiang, Y. Liang, J. Li, H. Ding, and L. Zhu. Omnidirectional humanoid locomotion on stairs via unsafe stepping penalty and sparse lidar elevation mapping, 2026. URL https://arxiv.org/abs/2603.07928.

[29] X. Duan, Z. Zhuang, H. Zhao, and S. Schwertfeger. Playful doggybot: Learning agile and precise quadrupedal locomotion, 2025. URL https://arxiv.org/abs/2409.19920.

[30] Z. Su, Y. Gao, E. Lukas, Y. Li, J. Cai, F. Tulbah, F. Gao, C. Yu, Z. Li, Y. Wu, and K. Sreenath. Toward real-world cooperative and competitive soccer with quadrupedal robot teams, 2025. URL https://arxiv.org/abs/2505.13834.

[31] Y. Ji, G. B. Margolis, and P. Agrawal. Dribblebot: Dynamic legged manipulation in the wild, 2023. URL https://arxiv.org/abs/2304.01159.

[32] X. Huang, Z. Li, Y. Xiang, Y. Ni, Y. Chi, Y. Li, L. Yang, X. B. Peng, and K. Sreenath. Creating a dynamic quadrupedal robotic goalkeeper with reinforcement learning, 2022. URL https://arxiv.org/abs/2210.04435.

[33] Z. Su, B. Zhang, N. Rahmanian, Y. Gao, Q. Liao, C. Regan, K. Sreenath, and S. S. Sastry. Hitter: A humanoid table tennis robot via hierarchical planning and learning, 2025. URL https://arxiv.org/abs/2508.21043.

[34] G. Pan, Q. Ben, Z. Yuan, G. Jiang, Y. Ji, S. Li, J. Pang, H. Liu, and H. Xu. Roboduet: Learning a cooperative policy for whole-body legged loco-manipulation, 2025. URL https://arxiv.org/abs/2403.17367.

[35] X. Liu, B. Ma, C. Qi, Y. Ding, N. Xu, Zhaxizhuoma, G. Zhang, P. Chen, K. Liu, Z. Jia, C. Guan, Y. Mo, J. Liu, F. Gao, J. Zhong, B. Zhao, and X. Li. Mlm: Learning multi-task loco-manipulation whole-body control for quadruped robot with arm, 2025. URL https://arxiv.org/abs/2508.10538.

[36] T. Portela, A. Cramariuc, M. Mittal, and M. Hutter. Whole-body end-effector pose tracking, 2025. URL https://arxiv.org/abs/2409.16048.

[37] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, 2nd edition, 2003. ISBN 978-0-521-54051-3.

[38] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[39] S. Chen, Z. Wan, S. Yan, C. Zhang, W. Zhang, Q. Li, D. Zhang, and F. U. D. Farrukh. Slr: Learning quadruped locomotion without privileged information, 2024. URL https://arxiv.org/abs/2406.04835.

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

[41] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022. URL https://arxiv.org/abs/2109.11978.

[42] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pages 510–517. IEEE, 2015.

[43] Q. Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

[44] H. K. Cheng, S. W. Oh, B. Price, J.-Y. Lee, and A. Schwing. Putting the object back into video object segmentation, 2024. URL https://arxiv.org/abs/2310.12982.

A Hardware Setup and Task Workflow

Hardware Setup. Our hardware setup utilizes the open-source 3D-printed mounting brackets from [15] to firmly secure the camera and the end-effector. Specifically, as shown in Figure 4, an overhead D435i camera is mounted with a fixed pitch angle. The end-effector itself is an ultra-low-cost ($< \$20$) servo-driven two-finger gripper.

Figure 4: **Hardware Setup.** The overhead D435i camera is mounted at a fixed pitch alongside an ultra-low-cost servo-driven two-finger gripper using brackets [15].

Task Workflow. As illustrated in Figure 5, during the pick-and-place evaluation, the robot is initially teleoperated to scan the environment and record the approximate world coordinates of all target objects. The autonomous system then operates in a loop: it utilizes these global coordinates to orient until the specific target enters the visual field, which subsequently triggers the visual-based closed-loop policy. We deliberately omit complex navigation routing, as it falls outside the core scope of evaluating the visuo-motor manipulation policy.

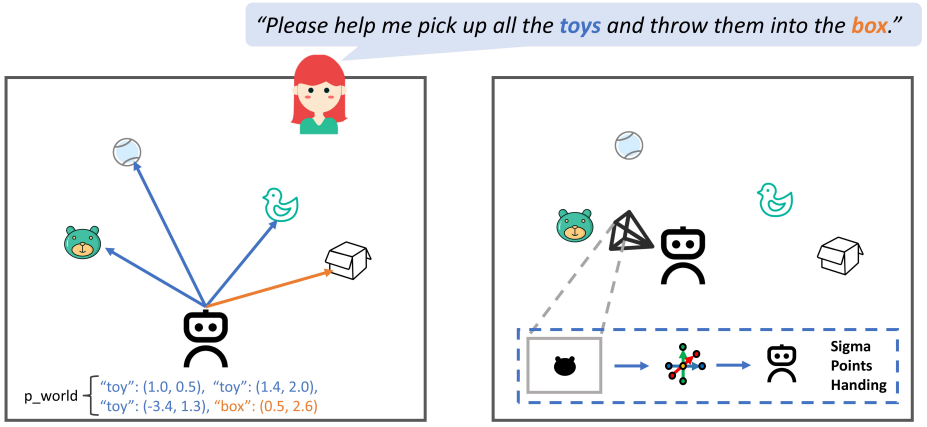

Figure 5: Overview of the task workflow. The robot first scans for approximate global coordinates, which are then used to orient the robot until the target is within the visual field to trigger the autonomous policy.

B Comprehensive Formulation of the Ego-Centric KF

To provide a complete and systematic mathematical overview, this section details the state definition, process model, and measurement update of our ego-centric Kalman Filter.

1. State Representation:
The filter tracks the spatial state of each Sigma Point $\mathbf{s}_j$ ($j \in \{0, \ldots, 6\}$) independently. The system

state is defined directly in the current camera frame $\{C_t\}$ as a 6D vector encompassing its 3D position and relative velocity:

$$ {}^{C_t}\mathbf{x}_{j,t} = \begin{bmatrix} {}^{C_t}\mathbf{s}_{j,t} \\ {}^{C_t}\mathbf{v}_{j,t} \end{bmatrix} \in \mathbb{R}^6 \tag{4} $$

2. Process Model and Ego-Motion Compensation:
The state transition is decoupled into point motion prediction and camera ego-motion compensation. First, a linear kinematic model predicts the point's displacement relative to the previous frame $\{C_{t-1}\}$ over a time step Δt, where Δt denotes the inter-frame interval:

$$ {}^{C_{t-1}}\mathbf{s}_{j,t}^{-} = {}^{C_{t-1}}\mathbf{s}_{j,t-1} + {}^{C_{t-1}}\mathbf{v}_{j,t-1}\Delta t \tag{5} $$

Subsequently, we rectify the shifts in the sensory frame caused by the robot's base movement. Using the relative transformation ${}^{C_t}\mathbf{T}_{C_{t-1}} \in SE(3)$ provided by the Visual Odometry (VO), we extract its rotational component ${}^{C_t}\mathbf{R}_{C_{t-1}} \in SO(3)$ and translational component ${}^{C_t}\mathbf{t}_{C_{t-1}} \in \mathbb{R}^3$. The *a priori* state estimate $\hat{\mathbf{x}}_{j,t}^{-}$ mapped to the new camera perspective $\{C_t\}$ is formulated as:

$$ {}^{C_t}\hat{\mathbf{s}}_{j,t} = {}^{C_t}\mathbf{R}_{C_{t-1}}\,{}^{C_{t-1}}\mathbf{s}_{j,t}^{-} + {}^{C_t}\mathbf{t}_{C_{t-1}}, \quad {}^{C_t}\hat{\mathbf{v}}_{j,t} = {}^{C_t}\mathbf{R}_{C_{t-1}}\,{}^{C_{t-1}}\mathbf{v}_{j,t-1} \tag{6} $$

3. Measurement Model and Dynamic Update:
When a visual observation is available, we extract the empirical 3D position by back-projecting the segmented image pixels, denoting this spatial observation as $\mathbf{z}_{j,t} \in \mathbb{R}^3$. Because the measurement space explicitly isolates the positional subspace of the full state vector, the observation model is strictly linear:

$$ \mathbf{z}_{j,t} = \mathbf{H}\,{}^{C_t}\mathbf{x}_{j,t} + \boldsymbol{\nu}_t, \quad \text{with} \quad \mathbf{H} = [\mathbf{I}_{3\times 3} \quad \mathbf{0}_{3\times 3}] \tag{7} $$

where $\boldsymbol{\nu}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ represents the measurement noise.

To handle depth-dependent projection errors in physical deployments, the measurement covariance matrix $\mathbf{R}_t$ is dynamically scaled relative to the estimated Z-axis depth (i.e., the distance along the optical axis from the camera to the observed point) of the observed point:

$$ \mathbf{R}_t = \text{diag}(\sigma_X^2, \sigma_Y^2, \sigma_Z^2), \quad \text{where} \quad \sigma_X = \frac{Z}{f_x}\sigma_u, \quad \sigma_Y = \frac{Z}{f_y}\sigma_v, \quad \sigma_Z = \sigma_z \tag{8} $$

with f_x and f_y denoting the intrinsic camera focal lengths (in pixels), σ_u and σ_v representing the pixel measurement uncertainties (in pixels), and σ_z being a constant depth uncertainty (in meters). This dynamic scaling explicitly degrades the filter's trust in distant, noisy observations.

Finally, the posterior state estimate is optimally corrected via the Kalman gain $\mathbf{K}_t$:

$$ \mathbf{K}_t = \mathbf{P}_t^{-}\mathbf{H}^T(\mathbf{H}\mathbf{P}_t^{-}\mathbf{H}^T + \mathbf{R}_t)^{-1} \tag{9} $$

$$ \hat{\mathbf{x}}_{j,t}^{+} = \hat{\mathbf{x}}_{j,t}^{-} + \mathbf{K}_t(\mathbf{z}_{j,t} - \mathbf{H}\hat{\mathbf{x}}_{j,t}^{-}) \tag{10} $$

where $\mathbf{P}_t^{-}$ is the prior error covariance matrix updated during the prediction step.

C Geometric Alignment and Task Evaluation Criteria

This section defines the geometric alignment errors and terminal-state criteria used by both reward gating and ASC evaluation.

Alignment Error Formulation. Let $\mathbf{p}_e \in \mathbb{R}^3$ and $\boldsymbol{\Theta}_e \in (-\pi, \pi]^3$ denote the end-effector position and Euler angles (roll, pitch, yaw), and let $\mathbf{p}_{\text{opt}} \in \mathbb{R}^3$ and $\boldsymbol{\Theta}_{\text{opt}} \in (-\pi, \pi]^3$ denote the target optimal pose. The alignment errors are defined by weighted squared norms:

$$ e_{\text{pos}}^2 = (\mathbf{p}_e - \mathbf{p}_{\text{opt}})^\top \mathbf{W}_p(\mathbf{p}_e - \mathbf{p}_{\text{opt}}), \quad e_{\text{rot}}^2 = (\boldsymbol{\Theta}_e - \boldsymbol{\Theta}_{\text{opt}})^\top \mathbf{W}_r(\boldsymbol{\Theta}_e - \boldsymbol{\Theta}_{\text{opt}}) \tag{11} $$

where $\mathbf{W}_p, \mathbf{W}_r \in \mathbb{R}^{3\times 3}$ are diagonal matrices specifying per-axis relative weights.

Let $\mathbf{p}_{\text{hint}} \in \mathbb{R}^3$ be the intermediate hint pose. The cross-track error to the hint-to-optimal segment is

$$ d_{\text{path}} = \|\mathbf{p}_e - \text{Proj}(\mathbf{p}_e, \overline{\mathbf{p}_{\text{hint}}\mathbf{p}_{\text{opt}}})\|_2 \tag{12} $$

where $\mathrm{Proj}(\mathbf{p}_e, \overline{\mathbf{p}_{\text{hint}}\mathbf{p}_{\text{opt}}})$ denotes the orthogonal projection of $\mathbf{p}_e$ onto the line segment connecting $\mathbf{p}_{\text{hint}}$ and $\mathbf{p}_{\text{opt}}$.

Terminal State Boundaries. Define $\Delta\mathbf{p} = \mathbf{p}_e - \mathbf{p}_{\text{opt}}$ and $\Delta\boldsymbol{\Theta} = \boldsymbol{\Theta}_e - \boldsymbol{\Theta}_{\text{opt}}$. Success and failure indicators are written as

$$\mathbb{1}_{\text{success}} = \mathbb{1}(|\Delta p_x| < \epsilon_x,\ |\Delta p_y| < \epsilon_y,\ |\Delta\Theta_{\text{yaw}}| < \epsilon_{\text{yaw}},\ |\Delta\Theta_{\text{pitch}}| < \epsilon_{\text{pitch}})\,, \tag{13}$$

$$\mathbb{1}_{\text{fail}} = \mathbb{1}(\text{timeout} \wedge [|\Delta p_x| \geq \delta_x \vee |\Delta p_y| \geq \delta_y \vee |\Delta\Theta_{\text{yaw}}| \geq \delta_{\text{yaw}} \vee |\Delta\Theta_{\text{pitch}}| \geq \delta_{\text{pitch}}])\,. \tag{14}$$

These indicators are used to compute the ASC running success rate s and to gate terminal reward activation.

- *Success State ($\mathbb{1}_{success}$):* Activated when the end-effector simultaneously satisfies translational and rotational tolerances relative to the optimal pose.
- *Failure State ($\mathbb{1}_{fail}$):* Activated at timeout when at least one positional or rotational error exceeds its failure bound.

All threshold constants and related hyperparameters are listed in Appendix G.

D Reward Function

To avoid complex heuristic shaping, the reward combines geometric path-following, sparse indicator terms, and stability penalties. Task-level terms guide hint-based tracking, terminal alignment, and field-of-view consistency, while regularization terms suppress wobbling and aggressive actions. We use a unified shaping term $E(\cdot) = \exp(-\|\cdot\|^2/\sigma_{\text{track}})$. Detailed geometric definitions are provided in Appendix C.

Table 2: Reward function components used in training.

Category	Term	Expression	Weight
Task & Auxiliary Constraints	r_{hint}	$E(d_{\text{path}}) \cdot E(e_{\text{rot}}) \cdot (1 + kE(e_{\text{pos}}))$	0.4
	r_{opt}	$\mathbb{1}_{\text{success}} \cdot E(e_{\text{pos}}) \cdot E(e_{\text{rot}}) \cdot E(\mathbf{v}_{\text{base}})$	20.0
	r_{miss}	$\mathbb{1}_{\text{out\_FOV}}$	-0.1
Stability & Control Regularizations	r_{roll}	g_y^2	-2.0
	r_{ang}	$\|\boldsymbol{\omega}_{xy}\|^2$	-0.1
	r_{smooth}	$\|\mathbf{a}_t - \mathbf{a}_{t-1}\|^2$	-0.01
	r_{limit}	$\|\mathbf{a}_{\text{clip}} - \mathbf{a}\|^2$	-0.1

E Kalman Filter Configuration

Table 3: Ego-centric Kalman filter parameters.

Category	Parameter	Value
Process Noise ($\mathbf{Q}$)	Position variance	1.0×10^{-6} m^2
	Velocity variance	1.0×10^{-5} m^2/s^2
Measurement Noise ($\mathbf{R}$)	Pixel std. dev. (σ_u, σ_v)	20.0 px
	Depth base std. dev. (σ_z)	0.05 m
State Initialization ($\mathbf{P}_0$)	Initial pos. variance	1.0×10^{-2} m^2
	Initial vel. variance	1.0×10^{-1} m^2/s^2

F Domain Randomization

Table 4: Domain randomization and observation noise parameters. $\mathcal{U}(a, b)$ denotes uniform distribution and $\mathcal{N}(0, \sigma)$ denotes zero-mean Gaussian noise.

Category	**Parameter**	**Distribution / Range**
Dynamics & Geometry	Ground friction	$\mathcal{U}(0.2, 5.0)$
	Restitution	$\mathcal{U}(0.0, 1.0)$
	Base added mass	$\mathcal{U}(-1.0, 2.0)$ kg
	Sigma Points physical scale (α)	$\mathcal{U}(1.0, 1.5)$
Camera Extrinsics	Translation (x, z)	$\mathcal{U}(-0.02, 0.02)$ m
	Translation (y)	$\mathcal{U}(-0.005, 0.005)$ m
	Orientation (roll, yaw)	$\mathcal{U}(-0.5^\circ, 0.5^\circ)$
	Orientation (pitch)	$\mathcal{U}(-2.0^\circ, 2.0^\circ)$
Observation Noise	Perception delay	$\mathcal{U}(0, 50)$ ms
	Base linear velocity	$\mathcal{N}(0, 0.1)$ m/s
	Base angular velocity	$\mathcal{N}(0, 0.1)$ rad/s
	Projected gravity	$\mathcal{N}(0, 0.1)$
	Sigma Points scale noise	$\mathcal{N}(0, 0.1)$
	Sigma Points rotation noise	$\mathcal{N}(0, 0.1)$ rad

G Training Configuration

Table 5: PPO Training and Network Configurations

Category	Parameter	Value
RL Hyperparameters (PPO)	Number of environments	4096
	Batch size	32768
	Minibatch size	8192
	Learning rate	3.0×10^{-4}
	Discount factor (γ)	0.99
	GAE parameter (λ)	0.95
	PPO clip range	0.2
	Entropy coefficient	0.01
NN Architectures	Actor MLP [dims]	[512, 256, 128]
	Critic MLP [dims]	[512, 256, 128]
	TCN Encoder [channels]	[32, 32, 32]
	TCN kernel size	5
	TCN dropout	0.1
	Activation function	ELU
	Short-horizon frames (H_{short})	5
	Long-horizon frames (H_{long})	10
Active Sampling & Blind-Spot	Curriculum threshold (s_{thresh})	0.15
	Transition decay rate (λ_{ASC})	5.0
	Initial feeding prob. ($p_{\text{near-optimal,start}}$)	0.8
	Final feeding prob. ($p_{\text{near-optimal,end}}$)	0.1
	Initial replay prob. ($p_{\text{failure-replay,start}}$)	0.2
	Final replay prob. ($p_{\text{failure-replay,end}}$)	0.5
	Drift standard deviation (σ_{drift}) [m]	0.01
	Max positional drift ($d_{\max}$) [m]	0.10
Terminal Criteria	Success bound: ϵ_x, ϵ_y [m]	0.05, 0.03
	Success yaw bound: ϵ_{yaw} [rad]	0.10
	Success pitch bound: ϵ_{pitch} [rad]	0.15
	Failure bound: δ_x, δ_y [m]	0.10, 0.10
	Failure yaw bound: δ_{yaw} [rad]	0.20
	Failure pitch bound: δ_{pitch} [rad]	0.20
Rewards & Actions	Tracking sigma: σ_{track}	0.04
	Hint-to-goal gain: k	1.0
	Action clip (v_x, v_y, ω_z) [m/s]	[-0.5, 0.5]
	Action clip (pitch) [rad]	[-π/6, π/6]