2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Sight Over Site: Perception-Aware Reinforcement Learning for Efficient Robotic Inspection

Richard Kuhlmann[1,*], Jakob Wolfram[1,*], Boyang Sun[2], Jiaxu Xing[3], Davide Scaramuzza[3], Marc Pollefeys[2,4], Cesar Cadena[1]

***Abstract*— Autonomous inspection is a central problem in robotics, with applications ranging from industrial monitoring to search-and-rescue. Traditionally, inspection has often been reduced to navigation tasks, where the objective is to reach a predefined location while avoiding obstacles. However, this formulation captures only part of the real inspection problem. In real-world environments, the inspection targets may become visible well before their exact coordinates are reached, making further movement both redundant and inefficient. What matters more for inspection is not simply arriving at the target's position, but positioning the robot at a viewpoint from which the target becomes observable. In this work, we revisit inspection from a perception-aware perspective. We propose an end-to-end reinforcement learning framework that explicitly incorporates target visibility as the primary objective, enabling the robot to find the shortest trajectory that guarantees visual contact with the target without relying on a map. The learned policy leverages both perceptual and proprioceptive sensing and is trained entirely in simulation, before being deployed to a real-world robot. We further develop an algorithm to compute ground-truth shortest inspection paths, which provides a reference for evaluation. Through extensive experiments, we show that our method outperforms existing classical and learning-based navigation approaches, yielding more efficient inspection trajectories in both simulated and real-world settings.**

The project is available at https://sight-over-site.github.io/

I. Introduction

Many real-world robotic applications fundamentally require a robot to obtain visual access to a target. Examples include inspecting devices [1], [2], reading measurement displays, detecting visual signals in industrial settings [3], tracking objects and people for service robots, enabling human-robot collaboration [4], [5], and supporting search-and-rescue operations [6], [7]. The core objective in these scenarios is similar and can be collectively described as autonomous inspection [8]–[10].

Despite its importance, inspection has received limited attention as a standalone objective. Most existing approaches treat inherently inspection-driven tasks as navigation problems, where the focus is on reaching a specific location [11]–[15], rather than explicitly optimizing for target visibility. In such cases, even though the goal is to observe the target, the robot primarily focuses on moving closer to it. Navigation-based approaches can occasionally succeed in inspection tasks, as proximity to the target increases the likelihood of observing it. However, this strategy often leads to a sub-optimal behavior. As shown in 1, the most direct path to achieve visibility can be significantly shorter than the trajectory chosen by a well-trained navigation policy. This discrepancy arises because navigation approaches typically do not consider visibility in their decision-making process. In real-world scenarios, this limitation can cause significant inefficiencies, such as the need to sequentially visit multiple viewpoints in cluttered environments.

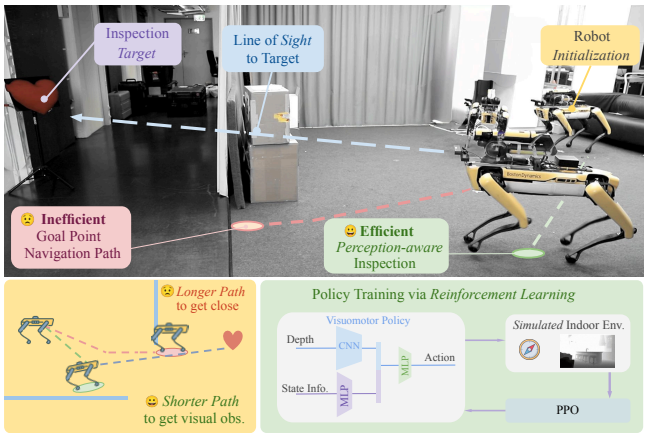

Fig. 1. To obtain visual access to a given target, the *inspection* policy (green path) results in a shorter trajectory compared to the *navigation* policy (red path). Our proposed RL-based policy (bottom right) takes egocentric depth input along with the target and robot state to achieve efficient inspection.

Inspired by these observations, we propose a perception-aware reinforcement learning (RL) framework for efficient robotic inspection. Our approach explicitly incorporates the visibility objective into the RL training process, encouraging the robot to obtain visibility of a target 3D location without relying on a map. The policy takes as input depth sensor readings, the relative distance and angle to the target, and the agent's last three actions, and outputs successive points of interest relative to the robot's position. A low-level controller then executes the corresponding joint commands to reach these points. We train the policy entirely in simulation, and it generalizes well to both unseen simulated environments and a real-world quadrupedal robot (Boston Dynamics Spot). Our policy outperforms current state-of-the-art point-goal navigation methods and demonstrates strong performance in

* These authors contributed equally to this work.
[1] Robotics Systems Lab, ETH Zurich, [2] Computer Vision and Geometry Group, ETH Zurich [3] Robotics and Perception Group, University of Zurich, [4] Microsoft Mixed Reality and AI Lab
Contact: rkuhlmann@ethz.ch

This work was supported by Huawei Tech R&D (UK) through a research funding agreement, by the Swiss National Science Foundation (SNSF) as part of project No.227617, and by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120732 (AUTOASSESS)

collision avoidance. Notably, the learned policy exhibits an emergent ability to balance navigation and inspection. When the target is far away, it prioritizes navigating towards the vicinity of the goal. Once nearby, it shifts focus towards reaching a feasible viewpoint as efficiently as possible. Contributions are as follows:

- A comprehensive definition and formulation of the perceptive inspection task, along with a modeling framework for evaluating the inspection process.
- A map-free, RL-based policy that leverages egocentric depth input and the relative pose to the target for efficient autonomous inspection.
- Quantitative evaluation of the proposed system against a state-of-the-art RL-based navigation approach, conducted in both simulation and real-world settings.

II. Related Work

Perception-aware navigation problems in unknown environments have been studied extensively in the past [13], [16], [17]. This general class of problems includes a range of tasks, such as exploration in unknown spaces [18]–[20], navigating to a specified location [11], [14], [15], or reaching specific objects or semantic regions [21], [22]. To address these different tasks, a variety of methods have been developed. A major line of work focuses on gradually constructing and updating a global map, which is then used to guide planning – either by selecting informative regions such as frontiers [23], [24] or by predicting next-best views to observe a target directly [10], [25]. More recently, with the rapid advancement of large-scale multi-modality models, several approaches have explored integrating high-level reasoning from these models into the mapping and decision-making process [26]–[29]. However, most existing methods focus on reaching a target location, often without explicitly considering whether the target becomes visible to the robot. Incorporating visibility into the objective, either through hand-designed information gain measures or queries to pre-trained language models, remains an open research question. Furthermore, to achieve robustness, many of these approaches rely on maintaining an explicit global map or long-term memory.

In parallel, another line of work leverages reinforcement learning to solve navigation tasks, avoiding modular system designs and taking advantage of large-scale, high-fidelity simulators. In particular, the point-goal navigation task has seen rapid progress using RL-based approaches. Policies trained on large-scale simulation data have demonstrated near-perfect performance [11]. Follow-up work has further improved performance in more challenging settings by developing better training schemes and network architectures [15], incorporating semantics [30], [31], extending to object-goal navigation [32], and enabling multi-robot coordination [33], [34]. Recent RL-based approaches have also improved generalization across tasks and environments, and reduced the sim-to-real gap [35], [36]. Despite this progress, few studies have investigated visibility as a core component of navigation.

Building on this body of work, we study the problem of *perception-aware inspection*. Specifically, we consider the task of navigating in an unknown, map-less environment to reach a vantage point from which the target is visible, using only egocentric depth input. In contrast to common navigation approaches, our formulation does not require the robot to reach the physical location of the target. Instead, the emphasis is placed on perception—finding a feasible viewpoint from which the target can be observed. This perspective provides a new lens for understanding and addressing inspection tasks in realistic, map-free settings.

III. Methodology

A. Task Definition

The objective in our *perceptive inspection* task is for a robotic agent to autonomously discover the shortest path to a vantage point from which a designated target becomes observable. A target is considered observable if it lies within the agent's field of view and the line of sight is free of occlusions caused by obstacles. In the classical point-goal navigation setting, the goal location is explicitly provided. In inspection, by contrast, the vantage point from which the target becomes visible is not known beforehand. The agent must therefore determine, from its sensory inputs, how to move to reveal the target. This makes the problem challenging, as it requires a tight integration of perception, planning, and control to actively search for informative viewpoints while avoiding collisions. In addition, motivated by prior work demonstrating that agents can operate effectively without relying on memory-intensive mapping [11], we frame the problem using end-to-end reinforcement learning. This choice avoids the overhead of explicit map construction, enabling agents to scale to large environments while still ensuring collision-free trajectories.

B. Formulating Evaluation Framework for Inspection

To evaluate the quality and efficiency of a robot's inspection trajectory, it is necessary to establish a ground-truth reference, namely the *shortest inspection path*, as illustrated in Alg. 1. We therefore propose an approach to determine this optimal path under full map observability in our perceptive inspection setting.

We define l as the shortest collision-free path from the robot's start position $\mathbf{S} \in \mathbb{R}^3$ to an inspection goal $\mathbf{G}_{\text{opt}} \in \mathbb{R}^3$ in the domain $D \subset \mathbb{R}^3$, from which the target $\mathbf{T} \in \mathbb{R}^3$ is observable. Given privileged access to a 2D top-down occupancy grid of the environment, the 3D domain D can be projected onto a planar subset $D_{\text{proj}} \subset \mathbb{R}^2$. The projected domain is partitioned into traversable space $D_{\text{free}} \subset D_{\text{proj}}$ and occupied space $D_{\text{occ}} = D_{\text{proj}} \setminus D_{\text{free}}$. Candidate inspection goal positions are grid cells $\mathbf{C} \in D_{\text{free}}$ that satisfy the following constraints: (i) *Traversability:* $\mathbf{C} \in D_{\text{free}}$. (ii) *Sensor range:* $\|\mathbf{T}_{\text{proj}} - \mathbf{C}\|_2 \leq \delta$, where $\mathbf{T}_{\text{proj}}$ is the 2D projection of the target. (iii) *Visibility:* The line of sight between $\mathbf{C}$ and $\mathbf{T}$ is unobstructed.

To determine visibility in 3D, we simulate the robot's depth sensor at grid cell $\mathbf{C}$. Let (u, v) be the pixel coordinates

Algorithm 1: Shortest Perceptive Inspection Path

Input: Occupancy grid D_{proj}, start $\mathbf{S}$, target $\mathbf{T}$, sensor range δ
Output: Shortest inspection path l
Project target: $\mathbf{T}_{\text{proj}} \leftarrow \text{proj}(\mathbf{T})$;
Define free space D_{free} from D_{proj} ;
$G \leftarrow \emptyset$; `// Candidate goal set`
for $\mathbf{C} \in D_{free}$ **do**
  if $\|\mathbf{T}_{proj} - \mathbf{C}\|_2 \leq \delta$ ***and*** *visible*$(\mathbf{C}, \mathbf{T})$ **then**
    $G \leftarrow G \cup \{\mathbf{C}\}$;
Run A* from $\mathbf{S}$ to goal set G, using heuristic $h(\mathbf{C}) = \min_{\mathbf{G} \in G} \|\mathbf{C} - \mathbf{G}\|_2$
$l \leftarrow$ resulting shortest path ;
return l

of the target's projection on the camera plane. Here we denote the measured depth at (u, v) by $d_{\text{depth}}(\mathbf{u})$ and the Euclidean distance from $\mathbf{C}$ to the target's projection by

$$d_{\text{target}} = \|\mathbf{T}_{\text{proj}} - \mathbf{C}\|_2. \tag{1}$$

The target is observable if and only if $d_{\text{depth}}(\mathbf{u}) \geq d_{\text{target}} > 0$. Otherwise, the line of sight is occluded by an obstacle.

Since the robot's translational motion is constrained to the xy-plane, the shortest inspection path can be computed on the 2D occupancy grid. Let the set of candidate inspection goal cells be:

$$G = \{\mathbf{G} \in D_{\text{free}} \mid \mathbf{G} \text{ satisfies constraints (i)–(iii)}\}.$$

We then apply the A* search algorithm [37] to find the shortest path from $\mathbf{S}$ to any $\mathbf{G} \in G$. The heuristic function is

$$h(\mathbf{C}) = \min_{\mathbf{G} \in G} \|\mathbf{C} - \mathbf{G}\|_2, \tag{2}$$

i.e., the Euclidean distance from the current cell $\mathbf{C}$ to the nearest candidate goal. When A* expands a cell in G, the corresponding shortest inspection path l is obtained. Thus, a single execution of A* is sufficient to determine the optimal path for a given start-target pair.

However, following this approach to use the ground-truth path as a reward signal in an RL training pipeline is impractical, since the set of candidate inspection goal points G changes every time the agent moves. This requires repeatedly running A* search on the occupancy grid along with visibility checks for the candidate goal cells. Therefore, in this work, we primarily employ the shortest perceptive inspection path for benchmarking. For RL policy training, instead of inefficiently recomputing ground-truth paths, we incorporate the same underlying principle into the environment and reward design to ensure efficient policy optimization, as detailed in the following subsections.

C. Environment Setup

From the environment setup, we follow several design choices established in prior PointGoal Navigation works [11], [13], [38]. At the beginning of each episode, the agent is initialized at a random position in the domain D with a random orientation. At each time step k, the agent observes its relative position $\Delta\mathbf{p}_k$ and orientation $\Delta\theta_k$ with respect to the target, together with an ego-centric depth image $\boldsymbol{I}_k \in \mathbb{R}^{H \times W}$. Based on the state and perception, the agent selects its next action.

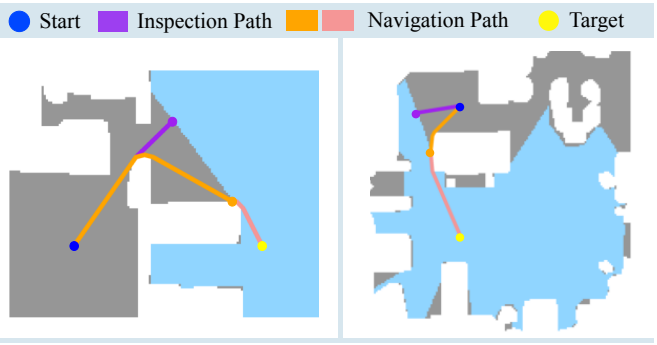

Fig. 2. Examples of shortest inspection path (purple) and the shortest navigation path, shown in orange until reaching a cell with target visibility and in pink thereafter. Traversable areas are depicted in gray, obstacles in white, and cells with target visibility in light blue.

D. Observation Space

At each time step k, the agent receives an observation $\mathbf{o}_k$. Formally, the observation $\mathbf{o}_k$ consists of four components: (i) the vector from the agent to the target $\Delta\mathbf{p}_k \in \mathbb{R}^2$, (ii) the relative orientation to the target $\Delta\theta_k \in [-\pi, \pi)$, (iii) an ego-centric depth image $\mathbf{I}_k \in \mathbb{R}^{H \times W}$, (iv) and the past three actions chosen by the agent, $\mathbf{u}_{k-1}, \mathbf{u}_{k-2}, \mathbf{u}_{k-3}$.

The position vector from the agent to the target $\Delta\mathbf{p}_k$ provides the agent with geometric context and serves as a high-level navigation cue. Importantly, unlike in PointGoal navigation, this information does not reveal the relative position to the true target of the inspection task, which is to identify the shortest path to a point in the domain D from which the target becomes observable, but the relative position to the target to inspect. The relative orientation $\Delta\theta_k$ is defined as the angular difference between the agent's heading direction and the direction of the vector $\Delta\mathbf{p}_k$. Since the agent is restricted to rotations around its vertical axis, this definition is sufficient and does not require considering the full $SO(3)$ rotation group. The ego-centric depth image $\mathbf{I}_k$ provides a local perceptual view of the environment from the agent's perspective. Depth is used instead of RGB to emphasize geometric structure, which is particularly important in indoor environments characterized by narrow hallways and frequent occlusions. Savva et al. [39] found depth observations to perform better than RGB for navigation settings. To incorporate temporal context and thereby enhance decision-making, the policy input is augmented with the three most recent actions executed by the agent.

E. Action Space

Given the sensor input at time step k, the agent learns to predict the next point of interest relative to its current pose in body frame. The agent's continuous action space U at time step k is represented by the vector $\mathbf{u}_k = [u_k^{(1)}, u_k^{(2)}, u_k^{(3)}]^\top$.

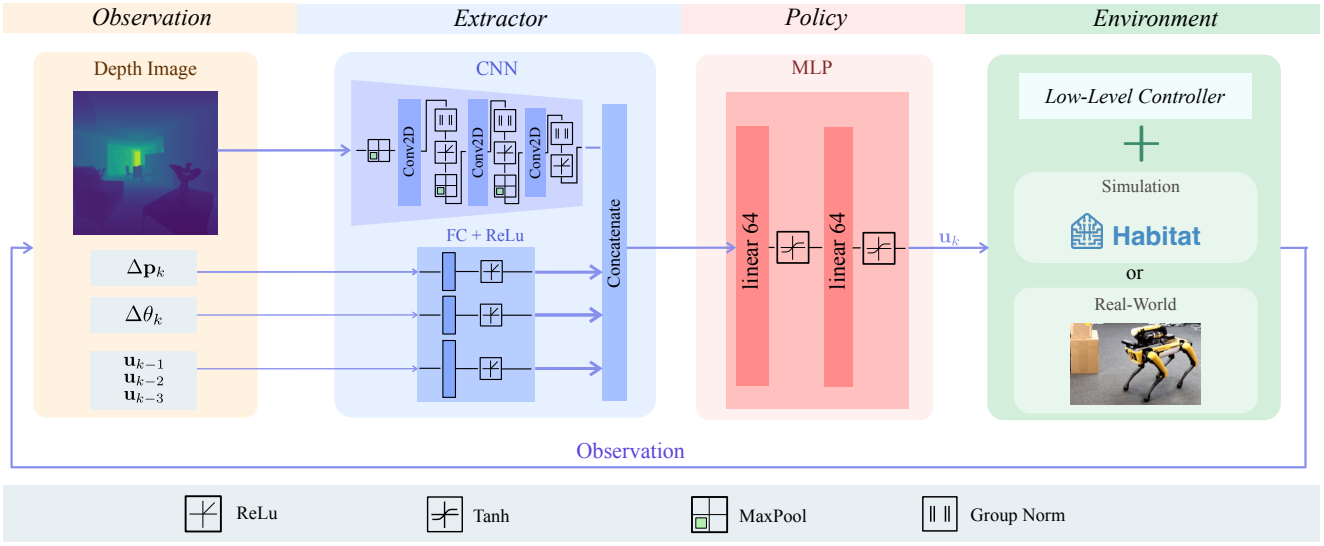

Fig. 3. Architecture of the pipeline for training and evaluation. A simulated environment or the robot's sensors provide the observations, which consist of the current ego-centric depth image $\mathbf{I}_k$, the position vector from the agent to the target $\Delta\mathbf{p}_k$, the relative orientation of the agent to the target $\Delta\theta_k$ and the past three actions the agent chose $\mathbf{u}_{k-1}, \mathbf{u}_{k-2}, \mathbf{u}_{k-3}$. Features are extracted from the depth image using a three-layer Convolutional Neural Network (CNN), while the remaining inputs are processed using Single-Layer Perceptrons (SLPs). The next action is selected from a continuous action space by a MLP policy and further executed by a low-level controller. The policy is trained using Proximal Policy Optimization (PPO).

Action $u_k^{(1)} \in [0,1]$ serves as an activation function and allows the agent to choose between moving forward and rotating in place:

$$u_k^{(1)} = \begin{cases} \text{move forward} & \text{if } u_k^{(1)} \leq 0.5. \\ \text{rotate in place} & \text{if } u_k^{(1)} > 0.5. \end{cases}$$

By using a continuous range for the activation value $u_k^{(1)}$ instead of a binary activation variable, we get insights into the uncertainty of the policy. If the policy is uncertain about whether to move forward or rotate in place at time step k, the value of $u_k^{(1)}$ remains close to 0.5.

The components $u_k^{(2)} \in [0,1]$ and $u_k^{(3)} \in [-1,1]$ specify the normalized magnitudes for forward translation and rotation, respectively. These values are subsequently scaled by their respective maximum limits, $s_{\max} = 0.35\,\text{m}$ for translation and $\phi_{\max} = \frac{\pi}{4}\,\text{rad}$ for rotation, to obtain the next point of interest. The decoupling of translation and rotation leads to more robust training and subsequently better policies.

In simulation, the agent's motion is governed by a linearized dynamics model. The agent's state at time step k is given by its position $\mathbf{p}_k = [x_k, y_k]^\top$ and orientation θ_k in the world frame. The update rule is defined as

$$\mathbf{p}_{k+1} = \begin{cases} \mathbf{p}_k + s_k \begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix} & \text{if } u_k^{(1)} \leq 0.5. \\ \mathbf{p}_k & \text{if } u_k^{(1)} > 0.5. \end{cases},$$

$$\theta_{k+1} = \begin{cases} \theta_k & \text{if } u_k^{(1)} \leq 0.5. \\ \theta_k + \phi_k & \text{if } u_k^{(1)} > 0.5. \end{cases},$$

where $s_k = u_k^{(2)} \cdot s_{\max}$ is the scaled forward translation and $\phi_k = u_k^{(3)} \cdot \phi_{\max}$ is the scaled rotation. Thus, at each time step k, the agent either moves forward in the direction of its current orientation or rotates in place, depending on the value of $u_k^{(1)}$.

In real-world deployment, a low-level controller is used to convert the next point of interest in the body frame into joint commands to be executed by the robot. Since our robot's motion during both training and deployment is not designed for high-speed operation, neglecting the implementation of a low-level controller does not affect real-world performance and even reduces training time.

F. Reward

Table I provides an overview of the reward terms. The term

$$\Delta\tilde{\mathbf{p}}_k = \frac{\Delta\mathbf{p}_k}{||\Delta\mathbf{p}_k||_2}$$

denotes the normalized position vector from the agent to the target at time step k. $\mathbf{n}_k$ represents the agent's heading vector at time step k. Both vectors are expressed in the world

TABLE I
OVERVIEW OF REWARD, CATEGORIZED INTO SPARSE AND DENSE TYPES.

Name	Value	Type
$r_{k,T}$	$C \cdot \left(1 + (\|\|\Delta\mathbf{p}_{k,\mathbf{G}_{\text{opt}}}\|\|_2/\sigma)^2\right)^{-1} \ldots$ $\cdot\mathbb{I}[t > T_{\text{episode}} - T_{\text{rew}}]$ -10	Sparse
$r_{k,\text{orient}}$	$\begin{cases} 0.2 \cdot \Delta\tilde{\mathbf{p}}_k \cdot \mathbf{n}_k & \text{if } \Delta\tilde{\mathbf{p}}_k \cdot \mathbf{n}_k > 0 \\ \Delta\tilde{\mathbf{p}}_k \cdot \mathbf{n}_k & \text{else} \end{cases}$	Dense
$r_{k,\text{nav}}$	$2.5 \cdot (\|\|\Delta\mathbf{p}_{k-1,\mathbf{G}_{\text{opt}}}\|\|_2 - \|\|\Delta\mathbf{p}_{k,\mathbf{G}_{\text{opt}}}\|\|_2)$	Dense
$r_{k,\text{move}}$	$\begin{cases} -0.5 & \text{if } \max_{j=1,2,3} \|\|\Delta\mathbf{p}_k - \Delta\mathbf{p}_{k-j}\|\|_2 < \frac{s_{max}}{4} \\ 0 & \text{else} \end{cases}$	Dense
$r_{k,\text{exp}}$	-0.05	Dense

frame. The term $||\Delta \mathbf{p}_{k,\mathbf{G}_{\text{opt}}}||_2 = ||\mathbf{p}_k - \mathbf{G}_{\text{opt}}||_2$ corresponds to the Euclidean distance between the agent and the optimal inspection goal point $\mathbf{G}_{\text{opt}} \in \mathbb{R}^2$. $\mathbf{G}_{\text{opt}}$ is computed during the dataset generation using the ground truth algorithm introduced in Section III-B.

We provide a terminal reward $r_{k,T}$ [40] at the end of an episode. The reward is computed based on the Euclidean distance $||\Delta \mathbf{p}_{k,\mathbf{G}_{\text{opt}}}||_2$ to the optimal inspection point at the end of the episode, balancing task completion with exploration. A penalty is imposed and the episode terminates in the case of a collisions and timeouts. These penalty terms encourage the agent to learn a policy that is both safe and efficient. The agent receives a reward $r_{k,\text{orient}}$ based on the relative orientation $\Delta\theta_k$ to the target, as illustrated in Figure 4. This reward is computed using the dot product of the agent's heading direction $\mathbf{n}_k$ and the normalized position vector from the agent to the target $\Delta \mathbf{p}_k$, yielding a continuous measure of alignment. An important advantage of this formulation is that the reward varies smoothly with the agent's orientation, providing continuous feedback rather than discrete penalties. In case the reward is positive, it is scaled down to prevent the agent from overfitting to this particular reward term and thus hinder its ability to navigate the environment.

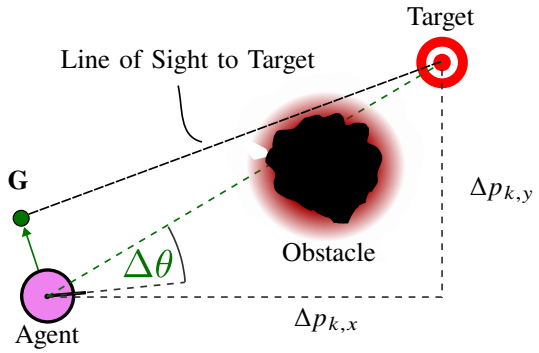

Fig. 4. In every step, the agent gets a reward based on the relative orientation to the target and the distance to the ground truth inspection point $\mathbf{G}_{opt}$. These rewards are indicated in green.

The reward term $r_{k,\text{nav}}$ is defined as a dense reward signal that encourages the agent to move towards the optimal inspection goal point $\mathbf{G}_{\text{opt}}$. It is computed by comparing the Euclidean distance from the previous position $\mathbf{p}_{k-1}$ to $\mathbf{G}_{\text{opt}}$ with the Euclidean distance from the current position $\mathbf{p}_k$ to $\mathbf{G}_{\text{opt}}$. Hence, if the agent chooses an action that reduces the Euclidean distance to the optimal inspection goal point $\mathbf{G}_{\text{opt}}$, it receives a positive reward. Conversely, if the agent's action increases the Euclidean distance to $\mathbf{G}_{\text{opt}}$, the agent receives a penalty. A key advantage of this formulation is its continuous nature, which provides smooth feedback at every time step k and thus facilitates stable learning.

During policy training, we observed that the agent often exhibits oscillatory behavior, repeatedly rotating in place until a timeout occurs. This behavior arises because the agent exploits the positive reward provided by $r_{k,\text{orient}}$ when it remains stationary while aligning with the target. To mitigate this issue, we introduce an additional penalty term, $r_{k,\text{move}}$, which penalizes the agent for remaining within a small neighborhood of its current position for more than three consecutive time steps. This term encourages the agent to make forward progress rather than indefinitely rotating in place, thereby promoting more efficient action choices. Moreover, we find that $r_{k,\text{move}}$ is particularly effective in helping the agent escape local minima, as it discourages stagnation at a single position.

Lastly, a small step penalty $r_{k,\text{exp}}$ is added to the total reward r_k to further incentivize exploration and encourage the agent to take a minimal number of actions. Thus, the total reward r_k provided at each time step k is described by the following function

$$r_k = \begin{cases} r_{k,T} & \text{if terminate at } k \\ r_{k,\text{orient}} + r_{k,\text{nav}} + r_{k,\text{move}} + r_{k,\text{exp}} & \text{else} \end{cases} \tag{3}$$

G. Model Architecture

Figure 3 provides an overview of the implemented policy architecture for our reinforcement learning pipeline. At every time step k, the agent receives an ego-centric depth image $\mathbf{I}_k$, the position vector from the agent to the target $\Delta \mathbf{p}_k$, the relative orientation of the agent to the target $\Delta\theta_k$ and the past three actions $\mathbf{u}_{k-1}, \mathbf{u}_{k-2}, \mathbf{u}_{k-3}$ as an input. A custom feature extractor is used to encode spatial and directional information from the agent's observations into a compact latent representation suitable for policy learning. A three-layer Convolutional Neural Network maps the depth image into a latent feature space, while the remaining inputs are transformed using Single-Layer Perceptrons (SLPs) into their respective latent representations. These feature vectors are then concatenated to form a unified latent state, which is provided to the policy network. Proximal Policy Optimization (PPO) [41] is employed to train the agent with a continuous action space due to its stability, sample efficiency, and ability to handle continuous action and observation spaces.

IV. Experiment and Result

A. Experimental Setup

We use the Habitat simulator [39] with the Gibson dataset [42] to train and evaluate our policy in realistic indoor environments. We focus on short-range inspection scenarios with start-to-target distances sampled uniformly between 3.5 and 4.5 meters. This range reflects typical indoor layouts and allows meaningful comparison between inspection and navigation behaviors. Specifically, the start ($\mathbf{S}_i$) and target ($\mathbf{T}_i$) positions for each episode are sampled uniformly from the traversable space $D_{\text{free}} \subset D$, constrained by:

$$3.5\text{m} \leq ||\mathbf{S}_{i,\text{proj}} - \mathbf{T}_{i,\text{proj}}||_2 \leq 4.5\text{m},$$

where $||\cdot||_2$ denotes Euclidean distance on the 2D floor plane.

To ensure diverse yet non-trivial episodes, we apply rejection sampling to keep the geodesic-to-Euclidean distance ratio within $[1.1, 1.5]$, following standard metrics used in

TABLE II

QUALITATIVE RESULTS UNDER: (I) COLLISIONS AND WALL-SLIDING ENABLED (C&S), (II) COLLISIONS ENABLED, WALL-SLIDING DISABLED (C&NS), AND (III) BOTH DISABLED (NC&NS).

Method	SR (C&S)	SPL (C&S)	SR (C&NS)	SPL (C&NS)	SR (NC&NS)	SPL (NC&NS)
Random	20.67%	0.0628	15.87%	0.0302	5.53%	0.0264
Obstacle Avoider	13.7%	0.1017	13.7%	0.1017	13.7%	0.1017
DD-PPO	**93.51%**	0.6170	75.72%	0.3397	19.47%	0.1291
Ours	90.38%	**0.6735**	**85.82%**	**0.6440**	**79.57%**	**0.6023**

navigation [39]. We analyze the shortest possible inspection path in each episode and find that agents travel, on average, 2.1 meters before gaining visibility of the target. In contrast, following the shortest A*-based navigation path until the target becomes visible results in an average of 2.45 meters. This supports our hypothesis that explicitly optimizing for visibility leads to more efficient inspection trajectories.

Training was conducted on an NVIDIA GeForce RTX 4090 for a total of 10 million time steps using 10 parallel environments, resulting in a wall-clock time of about two hours.

We test our policy on 400+ episodes across 14 previously unseen environments and compare it against three baselines: (1) a random policy, where the actions are sampled uniformly from the action space, (2) an obstacle-avoidance policy inspired by the Goal Follower baseline [39], and (3) DD-PPO [11], a state-of-the-art RL-based point-goal navigation policy. We picked a pre-trained version, that is trained on the Gibson-4+ subset, which is identical to our training set. Comparisons to classical search algorithms are omitted, as they require map knowledge, and our method explicitly is designed for map-free settings. We evaluate performance using two common metrics for navigation [43]:

- **Success Rate (SR)**: The percentage of episodes where the agent successfully observes the target, defined as:

$$\text{SR} = \frac{1}{N}\sum_{i=1}^{N} s_i, \tag{4}$$

  where $s_i = 1$ if the target is visible in episode i, and 0 otherwise.
- **Success-weighted Path Length (SPL)**:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} s_i \ \frac{l_i}{p_i}, \tag{5}$$

  where l_i is the shortest inspection path and p_i is the agent's actual path length.

To ensure fairness, all episodes terminate once the agent gains clear visual access to the target. Visibility is determined by projecting the 3D target location onto the camera plane and comparing its depth value to the agent's range, following the observability condition introduced in Section III-B.

B. Simulation Results

Table II presents the quantitative results across three evaluation settings: (i) collisions and wall-sliding enabled, (ii) collisions enabled but wall-sliding disabled, and (iii) neither collisions nor sliding allowed. The latter two reflect more realistic deployment scenarios, with the third setting being the most desirable. The first setting is specific to the simulator but is included to allow fair comparison with DD-PPO, which was trained under those conditions.

In the setting with both collisions and wall-sliding enabled, DD-PPO achieves a higher SR than our policy. However, our method yields a higher SPL, which means that it produces more efficient inspection paths. Notably, the SPL score of our policy decreases to 0.5875 in this setting if the reward r_{nav}, which guides the agent to the ground truth inspection point, is removed. This indicates the importance of this reward term. In the second evaluation setting, where collisions are allowed but wall-sliding is disabled, DD-PPO's performance declines relative to the first scenario. This suggests that DD-PPO leverages simulator-specific sliding dynamics. In contrast, our policy demonstrates greater robustness, maintaining similar SR and SPL values to those observed in the first setting. Finally, in the third setting, where collisions are entirely prohibited, our policy continues to perform well, achieving a success rate of 79.57% and demonstrating strong collision avoidance capabilities. It is also worth noting that, unlike in PointGoal Navigation tasks where the agent must explicitly issue a *Terminate* action to indicate task completion, we define success based on whether the target becomes visible to the agent. This criterion can inflate the success rate of random policies, as success does not require deliberate stopping behavior.

Figure 5 presents two qualitative examples. Our policy demonstrates clear advantages by taking shorter and more efficient paths to inspect the target. Unlike DD-PPO, which tends to follow the shortest navigation path that often stays close to the boundaries of the free space and results in occluded views of the target, our policy avoids such behavior. As a result, DD-PPO frequently requires a longer trajectory before the target becomes visible. In contrast, our policy effectively balances navigation and visibility. It not only moves toward the vicinity of the target but also actively seeks locally nearby viewpoints that are more likely to offer a clear line of sight. Additionally, our policy exhibits strong

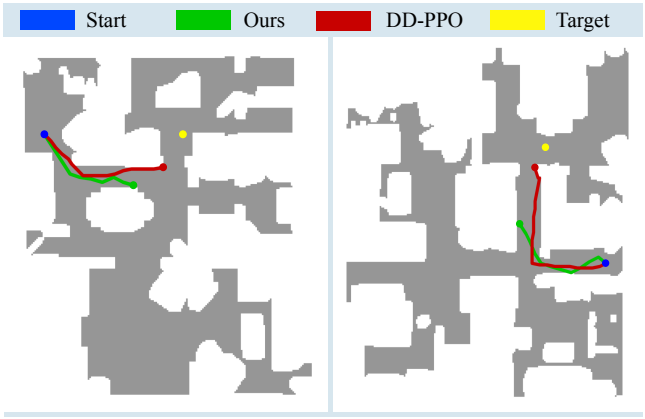

Fig. 5. Qualitative comparison between our policy and DD-PPO on test episodes. Traversable areas are shown in gray, while obstacles are marked in white. Our policy prioritizes maintaining visibility of the target, while DD-PPO focuses on minimizing navigation distance, often at the expense of target observability.

collision avoidance. Unlike DD-PPO, which often collides with and slides along walls, our agent navigates cleanly through cluttered environments.

C. Real-World Experiments

To evaluate the sim-to-real transferability of our policy, we deploy it on a Boston Dynamics Spot quadruped equipped with an NVIDIA Jetson Orin. For baseline comparison, we deploy the DD-PPO policy on the same platform. At the start of each test episode, a reference target point is specified. To provide the policy with the position vector from the agent to the target $\Delta\mathbf{p}_k$ and relative orientation $\Delta\theta_k$ at each inference step k, we leverage Spot's onboard odometry, which tracks its position relative to the episode's starting point. Depth images are captured using a ZED 2i camera and provided as input to the policy. Unlike the noise-free measurements available in the Habitat simulator, real-world measurements are inherently noisy. To mitigate its effect when checking the observability condition, we average the depth values within a 5×5 window centered on the projected goal point.

We conduct two experiments in which the robot is tasked with inspecting a designated target point. In both scenarios, the direct line of sight to the target is initially occluded:

1) Comparison of DD-PPO and our policy to highlight differences in inspection behavior.
2) Deployment of our policy under different initial conditions to evaluate its robustness.

In both cases, our policy demonstrates the intended *inspection behavior*, characterized by the robot minimizing unnecessary movement. A distinct difference between the behavior of DD-PPO and our policy becomes evident, as DD-PPO executes the shortest navigation path and our policy minimizes movement by finding a shorter path to a position from which the target can be observed. Notably, we deploy our policy zero-shot without retraining.

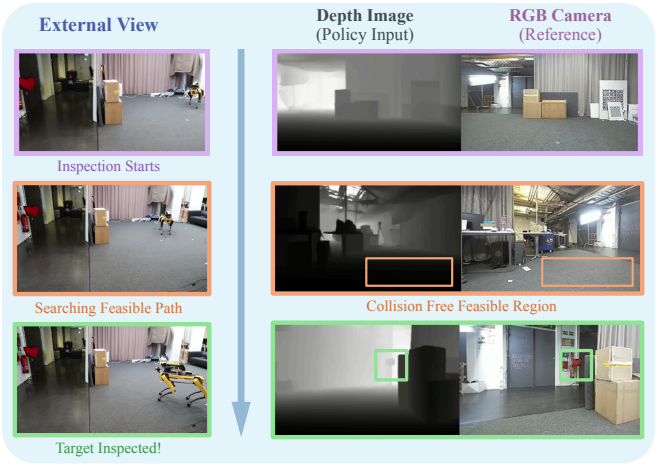

Fig. 6. Real-world experiments of our proposed system were conducted on a physical robot. On the left, we present external views showing how the robot gradually navigates to perceive the target without relying on explicit mapping. On the right, we visualize the onboard sensing configuration for both the depth and RGB cameras.

V. Conclusion

In this work, we focus on autonomous inspection, which requires a robot to obtain visual access to a target as efficiently as possible. Although this task is critical for many real-world applications, it is often treated as a navigation problem, leading to sub-optimal solutions. We begin by formalizing the inspection task and introducing a theoretical metric to evaluate inspection optimality. We then propose a reinforcement learning-based policy that explicitly incorporates visibility as a training objective. Our approach does not rely on any global map and is trained entirely in simulation. Despite this, it outperforms a state-of-the-art RL-based navigation policy in target inspection tasks. Finally, we demonstrate that our trained policy generalizes zero-shot to a real-world legged robot, showing strong performance in practical inspection scenarios. Currently, we assume that the target location is known a priori, which is unrealistic for fully autonomous inspection missions with unknown or dynamically emerging targets. Consequently, future work should extend the formulation toward a more general setting that jointly addresses target search and inspection.

References

[1] J. Xing, G. Cioffi, J. Hidalgo-Carrió, and D. Scaramuzza, "Autonomous power line inspection with drones via perception-aware mpc," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1086–1093.

[2] A. Ollero, A. Suarez, J. M. Marredo, G. Cioffi, R. Penicka, G. Vasiljevic, and A. Viguria, "Application of intelligent aerial robots to the inspection and maintenance of electrical power lines," *Robotics and Automation Solutions for Inspection and Maintenance in Critical Infrastructures; Now Publishers: Delft, The Netherlands*, 2024.

[3] D. Lattanzi and G. Miller, "Review of robotic infrastructure inspection systems," *Journal of Infrastructure Systems*, vol. 23, no. 3, p. 04017004, 2017.

[4] J. Chen, B. Sun, M. Pollefeys, and H. Blum, "A 3d mixed reality interface for human-robot teaming," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 11 327–11 333.

[5] M. Munaro and E. Menegatti, "Fast rgb-d people tracking for service robots," *Autonomous Robots*, vol. 37, pp. 227 – 242, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:18846188

[6] P. A. der Maur, B. Djambazi, Y. Haberthür, P. Hörmann, A. Kübler, M. Lustenberger, S. Sigrist, O. Vigen, J. Förster, F. Achermann, *et al.*, "Roboa: Construction and evaluation of a steerable vine robot for search and rescue applications," in *2021 IEEE 4th International Conference on Soft Robotics (RoboSoft)*. IEEE, 2021, pp. 15–20.

[7] S. Schwaiger, L. Muster, G. Novotny, M. Schebek, W. Wöber, S. Thalhammer, and C. Böhm, "Semi-autonomous mobile search and rescue robot for radiation disaster scenarios," *arXiv e-prints*, pp. arXiv–2406, 2024.

[8] A. Bircher, K. Alexis, U. Schwesinger, S. Omari, M. Burri, and R. Siegwart, "An incremental sampling-based approach to inspection planning: the rapidly exploring random tree of trees," *Robotica*, vol. 35, no. 6, p. 1327–1340, 2017.

[9] M. F. Ginting, D. D. Fan, S.-K. Kim, M. J. Kochenderfer, and A.-A. Agha-Mohammadi, "Semantic belief behavior graph: Enabling autonomous robot inspection in unknown environments," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 7604–7610.

[10] M. Dharmadhikari and K. Alexis, "Semantics-aware exploration and inspection path planning," *arXiv preprint arXiv:2303.07236*, 2023.

[11] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1gX8C4YPr

[12] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot," *arXiv preprint arXiv:2210.03370*, 2022.

[13] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," *arXiv preprint arXiv:2006.13171*, 2020.

[14] F. Yang, C. Cao, H. Zhu, J. Oh, and J. Zhang, "Far planner: Fast, attemptable route planner using dynamic visibility update," in *2022 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2022, pp. 9–16.

[15] F. Yang, C. Wang, C. Cadena, and M. Hutter, "iPlanner: Imperative Path Planning," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023.

[16] F. Bonin-Font, A. Ortiz, and G. Oliver, "Visual navigation for mobile robots: A survey," *J. Intell. Robotics Syst.*, vol. 53, no. 3, p. 263–296, Nov. 2008. [Online]. Available: https://doi.org/10.1007/s10846-008-9235-4

[17] T. Zhang, X. Hu, J. Xiao, and G. Zhang, "A survey of visual navigation: From geometry to embodied ai," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105036, 2022.

[18] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an mav," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 9570–9576.

[19] C. Cao, H. Zhu, H. Choset, and J. Zhang, "Tare: A hierarchical framework for efficiently exploring complex 3d environments." in *Robotics: Science and Systems*, vol. 5, 2021, p. 2.

[20] B. Sun, H. Chen, S. Leutenegger, C. Cadena, M. Pollefeys, and H. Blum, "Frontiernet: Learning visual cues to explore," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 6576–6583, 2025.

[21] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *Science Robotics*, vol. 8, no. 79, p. eadf6991, 2023.

[22] Y. Tao, Y. Wu, B. Li, F. Cladera, A. Zhou, D. Thakur, and V. Kumar, "Seer: Safe efficient exploration for aerial robots using learning to predict information gain," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 1235–1241.

[23] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, 1997, pp. 146–151.

[24] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "Poni: Potential functions for objectgoal navigation with interaction-free learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 890–18 900.

[25] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto, "An efficient sampling-based method for online informative path planning in unknown environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1500–1507, April 2020.

[26] J. Chen, G. Li, S. Kumar, B. Ghanem, and F. Yu, "How to not train your dragon: Training-free embodied object goal navigation with semantic frontiers," *arXiv preprint arXiv:2305.16925*, 2023.

[27] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "Vlfm: Vision-language frontier maps for zero-shot semantic navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 42–48.

[28] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "Vint: A foundation model for visual navigation," 2023. [Online]. Available: https://arxiv.org/abs/2306.14846

[29] H. Shah, J. Xing, N. Messikommer, B. Sun, M. Pollefeys, and D. Scaramuzza, "Foresightnav: Learning scene imagination for efficient exploration," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 5236–5245.

[30] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter, "Viplanner: Visual semantic imperative learning for local navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5243–5249.

[31] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[32] W. Xie, H. Jiang, Y. Zhu, J. Qian, and J. Xie, "Naviformer: A spatio-temporal context-aware transformer for object navigation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 14, 2025, pp. 14 708–14 716.

[33] S. Morad, A. Shankar, J. Blumenkamp, and A. Prorok, "Language-conditioned offline rl for multi-robot navigation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 14 984–14 991.

[34] M. Feng, V. Parimi, and B. Williams, "Safe multi-agent navigation guided by goal-conditioned safe reinforcement learning," *arXiv preprint arXiv:2502.17813*, 2025.

[35] W. Cai, J. Peng, Y. Yang, Y. Zhang, M. Wei, H. Wang, Y. Chen, T. Wang, and J. Pang, "Navdp: Learning sim-to-real navigation diffusion policy with privileged information guidance," *arXiv preprint arXiv:2505.08712*, 2025.

[36] S. Boiteau, F. Vanegas, J. Galvez-Serna, and F. Gonzalez, "Model-based rl decision-making for uavs operating in gnss-denied, degraded visibility conditions with limited sensor capabilities," *Drones*, vol. 9, no. 6, p. 410, 2025.

[37] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[38] R. Partsey, E. Wijmans, N. Yokoyama, O. Dobosevych, D. Batra, and O. Maksymets, "Is mapping necessary for realistic pointgoal navigation?" in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 17 232–17 241.

[39] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[40] T. He, C. Zhang, W. Xiao, G. He, C. Liu, and G. Shi, "Agile but safe: Learning collision-free high-speed legged locomotion," in *Robotics: Science and Systems (RSS)*, 2024.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[42] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[43] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," 2018. [Online]. Available: https://arxiv.org/abs/1807.06757