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

Robust Robot Navigation through Failure-Aversion Learning

Zhifeng Yu[1]*, Xuyang Li[1]*, Jianwu Fang[1], Guangliang Li[2], Jianru Xue[1†]

***Abstract*— Autonomous navigation in complex dynamic environments remains a fundamental challenge in robotics, and many reinforcement learning (RL) algorithms have demonstrated promising results, especially the on-policy ones. However, the inherent sample efficiency issue is still a fundamental problem to be solved. Methods integrating off-policy approaches into on-policy frameworks have been proposed to improve the sample efficiency by focusing on imitating the agent's past exemplary experiences while discarding less optimal ones. However, these methods overlook the valuable insights embedded within failures. Although some research has begun to explore learning from failures, it is usually done at a point-by-point level, ignoring the rich sequence context inherent in the trajectory. In this paper, we introduce DFPS-Nav, a training framework that utilizes Failure-Aversion Learning (FAL) to perform segmented, trend-based credit assignment, identifying both failure-inducing actions and valuable recovery behaviors within failed trajectories. We further improve successful imitation by adopting Prioritized Self-Imitation Learning (PSIL), which scores trajectories and prioritizes high-quality behaviors so that successful behaviors are reliably reproduced. Extensive simulation and real-world experiments demonstrate that using both FAL and PSIL to extract and refine information from the sequential context within trajectories, DFPS-Nav achieves up to 29.5% and 27% higher success rates in static and dynamic environments compared to the strong baseline method and is successfully applied in the real world. This work underscores how systematically deconstructing failures while prioritizing successes leads to more efficient and robust autonomous navigation.**

I. Introduction

Autonomous navigation is a core capability for unmanned ground vehicles (UGVs) to operate in real-world environments such as warehouses [1], public spaces [2], and disaster areas [3]. These environments are typically unstructured, crowded, and dynamic [4], requiring strategies that are both effective, robust, and adaptable [5]. Deep Reinforcement Learning (DRL) has been widely applied in autonomous navigation [6], as it allows agents to learn complex behaviors directly from sensorimotor interactions, reducing the need for manual engineering of complex rules [7].

This work is supported by the National Key Research and Development Program of China (Grant No. 2024YFE0210700), National Science Foundation of China (Grant No. 62273057), Outstanding Youth Foundation of Shaanxi Province (Grant No. 2025JC-JCQN-092), and Key Research and Development Program of Shaanxi (Grant No. 2025PT-ZCK-66).

[1]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China.

[2]College of Electronic Engineering, Ocean University of China, Qingdao, China.

* These authors contributed equally.

† Corresponding author

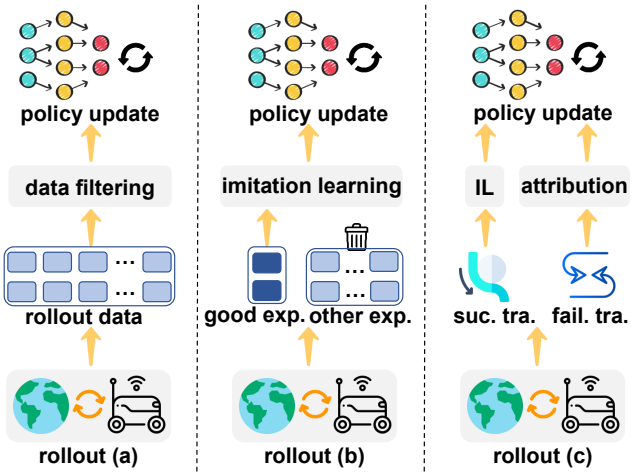

Fig. 1. Paradigm comparison. (a) On-policy updates driven by online evaluation (optional advantage filtering); (b) Self-imitation on good experiences, merged with the policy update, while other experiences are unused; (c) Ours: add segment-wise failure attribution on failed trajectories to retain/avoid key actions and refine the learning from good experience by introducing a trajectory-level quality-driven prioritization mechanism, followed by a joint update with success-priority imitation.

Among various DRL approaches, on-policy algorithms have been widely adopted for navigation due to their stable learning process and strong performance [6]. However, traditional on-policy reinforcement learning algorithms are hampered by their sample-inefficiency, as they discard entire batches of experience after just one gradient update [8], as illustrated conceptually in Fig. 1 (a). To address the challenge, some researchers integrate on-policy and off-policy [9] [10]. A representative example is Self-Imitation Learning (SIL) [9], which attempts to accelerate training by imitating the agent's own past good experience. However, these methods focus only on experiences of excellence, ignoring the valuable learning signals contained in failed attempts. These include identifying specific missteps that led to failure or recognizing corrective actions that nearly prevented it. Although TD-error [11] and advantage [9] are widely adopted to prioritize and replay high-impact state–action pairs, these scalar signals are not sufficient to capture the fine-grained distinctions between failure and recovery behaviors, as illustrated in Fig. 1 (b). While some works have begun to explore learning from failure [12] [13], they typically operate at a point-wise level, disregarding the rich sequential context that is inherent in the trajectory [14].

Therefore, we propose a training framework, **D**econstructing **F**ailures and **P**rioritizing **S**uccesses Asymmetric Learning Framework for Robot **Nav**igation (DFPS-Nav), which performs a segmented trend-based credit assignment analysis on failed trajectories to learn robust

avoidance and recovery behaviors in the Failure-Aversion Learning (FAL) module. Simultaneously, we refine the learning from successes by introducing a trajectory-level quality-driven prioritization mechanism on Prioritized Self-Imitation Learning (PSIL) that evaluates trajectories on metrics like efficiency and smoothness. By asymmetrically analyzing both successful and failed experiences at a global view (trajectory-level) and a local view (segment-level) instead of treating all data individually, we form our complete framework, as illustrated in Fig. 1 (c).

We evaluate DFPS-Nav using the VMAS [15] emulator in challenging random obstacle environments that incorporate static structured obstacles, H-shaped unstructured barriers, and dynamic obstacles. In addition, we conduct physical robot experiments to validate the applicability in the real world. The results of both simulation and physical experiments confirm that DFPS-Nav outperforms baseline methods in terms of success rate, collision avoidance, path quality, and convergence speed, demonstrating that DFPS-Nav is a robust and data-efficient framework for autonomous navigation. The main contributions of this paper are summarized as follows:

- We introduce a success-failure asymmetric learning framework (DFPS-Nav) that jointly leverages successful and failed experiences for safe and efficient navigation.
- We propose two complementary modules, PSIL and FAL, which realize trajectory-level prioritization and segment-level trend-based credit assignment, respectively.
- We evaluate DFPS-Nav in extensive simulation and physical robot experiments, and the results demonstrate its superior performance and robust generalization in complex environments.

II. Related Work

Our work builds upon several key areas in reinforcement learning and robot navigation.

A. Reinforcement Learning for Navigation

DRL has been widely applied to navigation tasks, from map-based planning to end-to-end sensorimotor control [16]. Algorithms such as DQN [17], A3C [18], and PPO [19] have been used to train agents to navigate in both static and dynamic environments. However, real-world robot navigation remains challenging for DRL. Sample inefficiency is a primary concern. The problem is especially pronounced in on-policy methods such as PPO [8]. PPO discards entire batches of collected experience after each gradient update. This leads to substantial data wastage. Such inefficiency severely limits feasibility for physical robot deployment, where data collection is costly and time-consuming [6]. Traditional DRL also struggles to learn robustly from sparse or ambiguous negative feedback. A simple binary reward (e.g., -1 for collision and $+1$ for reaching a goal) provides insufficient information for fine-grained credit assignment [14]. The agent then has difficulty discerning the specific actions or states that led to failure. This ambiguity can produce brittle policies. The learned behavior may become overly conservative. It may also converge to suboptimal strategies. To address these limitations, we introduce DFPS-Nav. The design targets sample inefficiency in on-policy methods and the lack of robust learning from negative feedback. Our framework integrates learning from both high-quality successful trajectories and information-rich failed trajectories. It moves beyond traditional step-level analysis. It leverages the sequential context embedded within the trajectory.

B. Augmenting On-Policy Reinforcement Learning with Experience Replay

To address the issue of sample efficiency, researchers have begun to explore the integration of the concept of off-policy into the on-policy framework [9] [20].

Learning from Successes via Self-Imitation Learning: Self-Imitation Learning (SIL) [9] is a representative method that accelerates the learning process by reusing past successful experiences collected by the agent. This approach allows an agent to imitate highlighted moments in past experience. It leads to faster mastery of effective strategies. SIL has proven effective in practice. Most implementations, however, treat all good experiences equally. They fail to distinguish between "good" and "better" experiences. Our Prioritized Self-Imitation Learning (PSIL) mechanism improves upon this limitation. It introduces a quality-based prioritization scheme. It ensures the agent focuses on imitating more efficient and smoother behaviors through trajectory-level evaluation. The proposed mechanism draws inspiration from Prioritized Experience Replay (PER) [21] in off-policy Q-learning. The formulation differs in a key way. We reformulate prioritization for on-policy optimization to enhance imitation learning.

Learning from Failures via Credit Assignment: Effectively handling failures is critical in robotics applications. The conventional approach is to apply a large negative reward upon entering a failure state, such as a collision. This only provides information on when a failure occurred. It does not explain why. More advanced methods, like Hindsight Experience Replay (HER) [12], learn from failures by reframing goals. This is more applicable to sparse-reward [20] [13], goal-oriented tasks. It is less suited to navigation tasks that require continuous risk avoidance. Another class of methods, safety-aware RL [22] [23], focuses on learning safety constraints to prevent failures entirely. This can overly restrict exploration. It can also lead to conservative policies. Our FAL module takes a unique approach by analyzing failed trajectories in segments. Rather than only penalizing the final outcome, it extracts informative positive cues from failed trials, such as moments of self-correction. This provides deeper and more insightful learning signals.

III. Methodology

A. Problem Formulation

We formulate the robot's navigation task as a Partially Observable Markov Decision Process (POMDP) [24] [25]. At each timestep t, the agent receives an observation o_t by concatenating the relative Cartesian coordinates to the

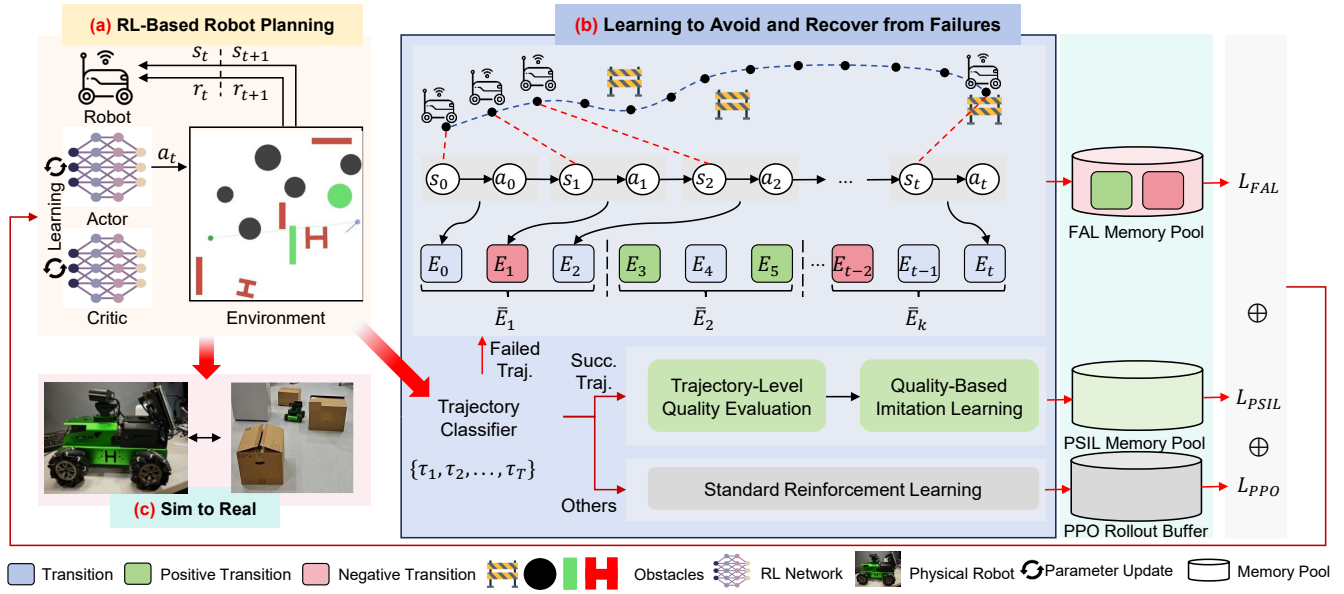

Fig. 2. **Overall architecture of DFPS-Nav with a progressive flow. (a) RL-based planning:** a standard on-policy loop establishes the baseline for navigation. **(b) Leveraging experiences:** collected trajectories are split into successful and failed sets; successful trajectories are stored in B_{PSIL} and drive success-priority imitation, while failed trajectories are stored in B_{FAL} and undergo segment-wise failure attribution. At each PPO iteration, minibatches sampled from B_{PSIL} and B_{FAL} are jointly used with the on-policy rollout minibatch to update the policy. **(c) Sim-to-Real:** the learned policy is deployed on a physical robot.

goal $p^g_{t,rel} \in \mathbb{R}^2$, its own planar velocity $v_t \in \mathbb{R}^2$, and the distance measurements from its N-beam LiDAR sensor, $l_t = \{l_{t,1}, ..., l_{t,N}\}$. Based on this observation, the agent selects a continuous action a_t, which is the 2D acceleration command (v_x, v_y) applied to the omnidirectional platform. The objective is to learn a policy $\pi(a_t|o_t)$ that maximizes the expected cumulative discounted return.

Inspired by DDPG-Nav [26], the reward function is designed to promote both efficiency and safety: $r_{t+1} = r_{target} + r_{obstacle} + r_{velocity} + r_{step} + r_{terminated}$, where the goal-shaping term is $r_{target} = k_1 \cdot \frac{d_{t-1}-d_t}{\Delta_{max}} + k_2 \cdot \frac{\theta_{dir}}{\pi} - k_3 \cdot \frac{\theta_v}{\pi}$, which encourages progress towards the goal and penalizes inefficient orientation. The obstacle penalty is $r_{obstacle} = r_{o1} + r_{o2}$, where $r_{o1} = -k_4 \cdot (\frac{1/d_t^{obs} - 1/\rho_0}{1/r_{safe} - 1/\rho_0})^{n_r}$ and $r_{o2} = -k_5 \cdot \frac{d_{t-1}^{obs} - d_t^{obs}}{\Delta_{max}}$, discouraging navigation too close to obstacles. To promote rapid movement, the velocity reward is $r_{velocity} = k_6 \frac{v}{v_{max}}$. A small constant penalty r_{step} is applied at each timestep to incentivize shorter paths. Finally, a large terminal reward, $r_{terminated}$, is assigned, which is R_{succ} for success (if $d_t < r_{safe}$) and R_{coll} for collision (if $d_t^{obs} < r_{safe}$).

Our framework is built on Proximal Policy Optimization (PPO) [8], an actor–critic algorithm that learns by estimating the advantage of taking an action in a given state. We use Generalized Advantage Estimation (GAE) [27] to compute the advantage function $\hat{A}_t$:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}, \quad \text{where } \delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t), \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor and $\lambda \in [0, 1]$ is a parameter to balance bias and variance in the advantage estimation. The term δ_t represents the Temporal-Difference (TD) error [11] at timestep t. The policy π_θ is then updated by maximizing a clipped objective:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between the new and old policies, and ϵ is a hyperparameter that defines the clipping range.

However, the standard PPO update mechanism has a key limitation in how the expectation $\hat{\mathbb{E}}_t$ in Eq. 2 is implemented. This expectation is approximated by averaging over mini-batches of transitions that are randomly sampled from the entire collection of experiences. Although this process stabilizes training, it treats each transition as an independent data point, thus discarding the rich sequential context embedded within a complete trajectory. We term this as the *trajectory-unaware* nature of the policy update step. To overcome this limitation and re-introduce trajectory-level context into the learning process, we propose Success-Failure Asymmetric Learning for Navigation (DFPS-Nav). As illustrated in Fig. 2, our framework labels each trajectory as successful, failed, or standard. It then routes the trajectory to a matching learning module, which helps with more precise credit assignment and more efficient use of all available data.

B. Failure-Aversion Learning via Trajectory-aware Attribution

A core component of our framework is Failure-Aversion Learning (FAL), a module designed to extract nuanced and

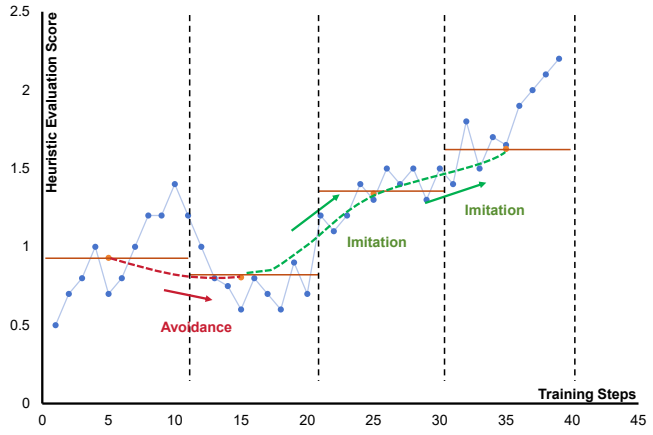

Fig. 3. **The schematic diagram of FAL.** A failed trajectory is scored over time (blue points) and partitioned into fixed-length segments separated by vertical dashed lines. For each segment, a horizontal line indicates its mean score; the difference between consecutive segment means gives the performance trend. Segments are then marked to imitate key actions (green upward arrow, rising trend) or avoid (red downward arrow, falling trend).

corrective signals from failed trajectories. The first step in this process is to establish a reliable performance metric to evaluate each action within such a trajectory. While Generalized Advantage Estimation (GAE, $\hat{A}_t$) is the standard for credit assignment due to its variance-reduction properties, its reliability hinges on a well-trained value function (the critic). However, failed trajectories often traverse out-of-distribution states where the agent has little experience. In these regions, the critic's value estimates, $V_\phi(s_t)$, can be inaccurate and systematically biased. A biased critic, in turn, can contaminate the GAE calculation, potentially yielding a misleading signal for what constituted a good or bad action.

To obtain a performance signal that is independent of the potentially unreliable critic in these failure regimes, we instead adopt a heuristic evaluation score $E(s_t, a_t)$ based on the raw, empirical future returns. While this Monte Carlo-based [28] metric inherently possesses higher variance, it provides an unbiased estimate of the true outcome of an action, free from the distortions of an inaccurate value function. We posit that for analyzing the macro-level *trend* of performance within a failed trajectory, this unbiased (though noisy) signal provides a more robust foundation.

For a given failed trajectory $\tau_{\text{fail}} = \{(s_t, a_t, r_t)\}_{t=0}^{T-1}$, we compute this score for each timestep t as:

$$E(s_t, a_t) = \frac{1}{T-t} \sum_{i=t}^{T-1} r_i, \tag{3}$$

where a higher $E(s_t, a_t)$ indicates that action a_t at state s_t led to a sequence of relatively higher returns, even if the overall trajectory ultimately ended in failure.

This score serves as the basis for our subsequent segment-level trend analysis. The failed trajectory is first partitioned into K non-overlapping temporal segments $\{S_k\}_{k=1}^{K}$. For each segment S_k, we compute the average evaluation score, $\bar{E}_k$. The performance trend is then measured by comparing the scores of adjacent segments, a process analogous to evaluating a function's derivative:

$$\Delta \bar{E}_k = \bar{E}_k - \bar{E}_{k-1}. \tag{4}$$

This raw trend signal $\Delta \bar{E}_k$ provides the foundation for our two-level guidance mechanism, which translates the macro-level trend into a fine-grained per-action learning signal. At the first level, we use the trend sign to provide coarse segment-level guidance. Specifically, an improving trend ($\Delta \bar{E}_k > 0$) suggests imitation because the segment may encode recovery behaviors, while a deteriorating trend ($\Delta \bar{E}_k < 0$) suggests avoidance (see Fig. 3). At the second level, we convert this coarse guidance into fine-grained supervision by selecting the most salient actions within each segment. In an imitation segment, we only select actions whose individual scores are above the segment's average ($E(s_t, a_t) > \bar{E}_k$), thus focusing on the best-performing behaviors. In an avoidance segment, we target the most detrimental actions by selecting those with scores below the segment's average ($E(s_t, a_t) < \bar{E}_k$).

This two-level guidance logic is then formalized into a single guidance weight $w_{\text{guide}}(s_t, a_t)$ which assigns an imitation signal (-1) or an avoidance signal (+1) exclusively to these filtered key actions, and all other actions receive a weight of zero:

$$w_{\text{guide}}\ (s_t, a_t) = \begin{cases} -1, & \text{if } \Delta \bar{E}_k > 0 \text{ and } E(s_t, a_t) > \bar{E}_k \\ +1, & \text{if } \Delta \bar{E}_k < 0 \text{ and } E(s_t, a_t) < \bar{E}_k \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

With the guidance weights for each key action determined, these informative samples are stored in a dedicated buffer B_{FAL}, and are used to update the policy via the following specialized loss function:

$$L_{FAL}(\theta) = \hat{\mathbb{E}}_{(s,a)\sim B_{FAL}} \left[w_{\text{guide}}(s, a) \cdot \log \pi_\theta(a|s) \right]. \tag{6}$$

In essence, this trend-based assignment paradigm allows the agent to learn robust recovery strategies by distinguishing not only between beneficial and harmful segments but also between the most critical actions within those segments. This perspective shifts the conventional paradigm from point-wise credit assignment to a more context-aware and trend-driven analysis, thus enhancing both the precision and robustness of policy optimization.

C. Trajectory-Level Structured Learning from Successful Experiences

To complement the fine-grained corrective learning from failures provided by FAL, our framework introduces Prioritized Self-Imitation Learning (PSIL), a module designed to extract and reinforce globally optimal behaviors from the most successful trajectories.

The core of PSIL lies in its quality-driven approach to self-imitation. It begins by evaluating each successful trajectory τ_{succ} with a holistic score $S(\tau_{\text{succ}})$ that combines both path

efficiency and smoothness:

$$S\left(\tau_{\text{succ}}\right) = w_{eff} \cdot \frac{1}{T} + w_{smooth} \cdot \left(\sum_{t=1}^{T-1} \|\mathbf{v}_t - \mathbf{v}_{t-1}\|_2 + \eta \right)^{-1}, \tag{7}$$

where T is the trajectory length, $\mathbf{v}_t$ is the agent's velocity at step t, and η is a small constant for numerical stability. Subsequently, we adapt the concept of Prioritized Replay [21]. All state-action pairs from a trajectory are stored in a buffer B_{PSIL} with their initial priority set proportional to the trajectory's holistic score $S(\tau_{\text{succ}})$. We then update the policy by minimizing a weighted Behavior Cloning (BC) loss [29], which encourages the policy to assign higher log-probability to the previous successful actions of the agent:

$$L_{PSIL}(\theta) = -\hat{\mathbb{E}}_{(s,a)\sim B_{PSIL}} \left[w_{PSIL} \cdot \log \pi_\theta(a \mid s) \right], \tag{8}$$

where w_{PSIL} are importance-sampling weights that correct for the non-uniform sampling distribution introduced by prioritization.

Through this combination of macro-structure learning and structured imitation, PSIL provides a trajectory-level constraint that complements the local and derivative-like learning of FAL, together forming a dual-level guidance paradigm.

D. Unified Training Objective

With the complementary guidance mechanisms of FAL and PSIL in place, we now integrate them with the PPO baseline into a unified training objective.

The final actor (policy) loss $L_{\text{actor}}(\theta)$ is defined as:

$$\begin{aligned} L_{\text{actor}}(\theta) = &- L^{CLIP}(\theta) - S(\theta) \\ &+ \lambda_{PSIL} L_{PSIL}(\theta) + \lambda_{FAL} L_{FAL}(\theta). \end{aligned} \tag{9}$$

This objective augments the standard PPO surrogate loss ($-L^{CLIP}$ defined in Eq. 2) and an entropy bonus ($S(\theta)$ for exploration) with the weighted losses from FAL (L_{FAL}, Eq. 6) and PSIL (L_{PSIL}, Eq. 8), controlled by coefficients λ_{PSIL} and λ_{FAL} respectively. Concurrently, the critic (value) network V_ϕ is trained using the standard mean squared error loss against the value targets R_t:

$$L_{\text{critic}}(\phi) = \hat{\mathbb{E}}_t \left[\left(V_\phi(s_t) - R_t \right)^2 \right]. \tag{10}$$

IV. Experiments

A. Environment Settings

Simulation Environment: We constructed a simulation environment for robot navigation based on the vectorized multi-agent simulator (VMAS) [15]. At the beginning of each episode, the robot's starting and ending points, as well as several dynamic and static obstacles, will be randomly generated. When the intelligent agent reaches the target point, collides, or reaches the maximum number of steps, the current episode will terminate. The relevant parameters are shown in the table.

Physical Environment: We conducted experiments on a physical robotic platform. The experiments were performed in a challenging indoor environment containing both static and dynamic obstacles, mirroring the complexity of our training scenarios. Our robotic platform is a mobile cart equipped with Mecanum wheels, enabling the omnidirectional movement assumed in our method's action space design. The robot is outfitted with a 360-point RPLidar A1 laser radar for environment perception. All onboard computation, including sensor data processing and policy inference, is handled by an Orin Nano 8G module. The policy is deployed to control the robot at a frequency of 10 Hz, with a maximum velocity capped at 0.5 m/s.

TABLE I
Environment configuration in simulation experiments.

Parameter	Value / Range
Map size	$[-1, 1] \times [-1, 1]$ (2m $\times$ 2m)
Agent radius	0.05 m
Static obstacles	$5 \sim 15$, radius $\in [0.15, 0.3]$ m
Trap obstacles	2, "H-shape" walls of width 0.08 m, length $[0.3, 0.5]$ m
Dynamic obstacles	$1 \sim 3$, random initial velocity $\in [-0.5, 0.5]$ m/s
Start position	Randomized on map
Goal position	Randomized on map
Episode horizon	$T = 1000$ steps ($\Delta t = 0.1$s)

Evaluation Metrics: To quantitatively evaluate the performance of navigation policies, we employ the following metrics. All metrics are averaged over 200 evaluation episodes. (1) **S**uccess **R**ate (**SR**): The percentage of episodes in which the agent reaches its goal without any collisions and within the maximum number of steps; (2) **C**ollision **R**ate (**CR**): The percentage of episodes that terminate due to the agent colliding with any obstacle or boundary wall; (3) **S**uccess-weighted by **P**ath **L**ength (**SPL**) [30]:

$$SPL = \frac{1}{N} \sum_{i=1}^{N} S_i \frac{L_i}{\max(P_i, L_i)},$$

where N is the number of testing environments, L_i is the distance between start and goal, P_i is the actual traveled distance, and $S_i \in \{0, 1\}$ indicates whether the agent successfully reached the goal. SPL measures both success and path efficiency, with higher values indicating more optimal navigation. (4) **S**uccess-weighted by **C**ompletion **T**ime (**SCT**) [31]:

$$SCT = \frac{1}{N} \sum_{i=1}^{N} S_i \frac{T_{opt,i}}{\max(T_{act,i}, T_{opt,i})},$$

where $T_{opt,i} = \frac{L_i}{v_{max}}$ is the optimal completion time based on maximum speed v_{max} and $T_{act,i}$ is the actual time taken. SCT evaluates success and time efficiency, with higher scores reflecting faster and more optimal navigation.

Baselines: To comprehensively evaluate our proposed method, we compare it against these representative baselines.

(1) **NavRL [19]:** An advanced single-agent navigation framework based on the PPO algorithm, although demonstrated on an unmanned aerial vehicle (UAV), its core DRL navigation principles are applicable to ground robots. To apply the NavRL framework to non-visual perception

scenarios, we replaced the original method's vision-based sensory input with direct environmental state information.

(2) **Self-Imitation Learning (SIL) [9]:** An algorithm designed to improve sample efficiency by learning from an agent's own past good experiences.

(3) **Proximal Policy Optimization (PPO) [8]:** A leading on-policy RL algorithm known for its stability and strong performance across a wide range of tasks.

Implementation Details: Our method is implemented based on the PPO algorithm. Both the actor and critic networks are implemented as Multi-Layer Perceptrons (MLPs) with three hidden layers of sizes [256, 256, 128]. For training, we utilize 600 parallel environments to collect data over a total of 1.2×10^8 timesteps. The experience replay buffers for Prioritized Self-Imitation Learning (PSIL) and Failure-Aversion Learning (FAL) are both set to a buffer size of 2×10^7, and 6×10^4 samples are drawn from each buffer in every training episode. The discount factor γ is set to 0.99, the GAE parameter λ is 0.95, and the minibatch size is 6×10^4. Notable hyperparameters include $k_1 = 5, k_2 = 3, k_3 = 1, k_4 = 3, k_5 = 2, k_6 = 0.1, \rho_0 = 0.3m, r_{safe} = 0.05m, n_r = 1.8, \Delta_{max} = 0.1m, v_{max} = 0.5m/s, w_{eff} = 1, w_{smooth} = 0.5, \lambda_{PSIL} = 1, \lambda_{FAL} = 0.1$. All experiments are conducted on a workstation equipped with an NVIDIA RTX 4060 GPU, Intel i5-14500 CPU, and 32GB RAM. Each training run requires approximately 8.5 GPU hours to converge.

B. Simulation Results and Analysis

As detailed in Tables II and III, our DFPS-Nav framework significantly outperforms all baseline methods in SR, achieving **92%** in static and **80%** in dynamic environments. While demonstrating a dominant SR, DFPS-Nav's SPL is lower than some baselines. We argue this is not a shortcoming but an intelligent strategic trade-off for safety in hazardous environments. When the agent confronts a potential collision with a dynamic obstacle or a challenging "H-shape" trap, the FAL module learns to take more cautious actions, such as decelerating, taking a wider detour, or executing a turn, as shown in Fig. 5. These longer paths naturally result in lower SPL scores but are crucial for prioritizing task completion. The severity of these traps is evidenced by the performance of NavRL: its SR jumps from 31.5% to 61.5% in the static environment once the "H-shape" walls are removed (Table II). This validates the necessity of our safety-oriented approach.

TABLE II

PERFORMANCE COMPARISON IN STATIC ENVIRONMENTS.

	Static Env.			
Benchmarks	SR↑	CR↓	SPL↑	SCT↑
PPO [8]	57%	15.5%	0.035	0.081
NavRL [19]	31.5%	**0.0%**	**0.068**	0.068
NavRL† [19]	61.5%	3.5%	0.040	0.070
PPO+SIL [9]	62.5%	22.0%	0.038	**0.087**
Ours (DFPS-Nav)	**92.0%**	7.5%	0.023	0.080

† Tested in an environment without "H-shape" trap obstacles.

TABLE III

PERFORMANCE COMPARISON IN DYNAMIC ENVIRONMENTS.

	Dynamic Env.			
Benchmarks	SR↑	CR↓	SPL↑	SCT↑
PPO [8]	53%	12%	0.038	**0.081**
NavRL [19]	30.5%	34.0%	**0.061**	0.070
NavRL† [19]	66.0%	**0.0%**	0.041	0.072
PPO+SIL [9]	53.0%	12.0%	0.038	0.079
Ours (DFPS-Nav)	**80.0%**	14.5%	0.028	**0.081**

† Tested in an environment without "H-shape" trap obstacles.

TABLE IV

THE PERFORMANCE OF FAL MODULE COMPARISON UNDER DIFFERENT SEGMENT LENGTHS IN STATIC ENVIRONMENT.

Seg Len.	SR↑	CR↓	SPL↑	SCT↑
5	**88%**	**7.5%**	0.023	0.077
10	80%	11.5%	**0.026**	**0.082**
15	81%	11.5%	0.022	0.076
20	80.5%	13%	0.023	0.075

C. Ablation Study

To isolate the individual contributions of our framework's two core modules: Prioritized Self-Imitation Learning (PSIL) and Failure-Aversion Learning (FAL), and to verify their synergistic effect, we conducted an ablation study. We compared the full DFPS-Nav model against variants containing only PSIL (PPO-PSIL) and only FAL (PPO-FAL), as well as the standard PPO algorithm. As shown in Table V and Table VI, compared to the standard PPO, the PSIL module improves the success rate from 57% to 80% in the static environment and from 53% to 66% in the dynamic environment. This indicates that PSIL effectively guides the policy toward more optimal behaviors by imitating high-quality trajectories. On the other hand, the FAL module drastically reduces the collision rate from 15.5% to 10.5% in the static environment while increasing the success rate to 84% and 56.5% in the static and dynamic environment. This confirms that the FAL module successfully enables the agent to learn hazard avoidance and recovery strategies through its segment-level trend analysis and credit assignment on failed trajectories.

What's more, the complete DFPS-Nav model which integrates both modules, performs the best in the success rate metric, achieving a 92% and 80% in the static and dynamic environment, far superior to any single-module variant. This result clearly demonstrates a powerful synergy between PSIL and FAL. PSIL provides integral guidance by evaluating global, trajectory-level patterns, while FAL provides derivative-like local correction by identifying critical turning points within failures. This combined paradigm of global-local learning, which leverages both successful and failed experiences, allows DFPS-Nav to achieve more efficient and safer navigation performance, thereby validating the rationale and superiority of our bi-constraint framework design.

Furthermore, to study the impact of the segment length for FAL, we compared the performance of FAL under different segment lengths, as shown in Table IV. While a segment length of 5 yields the highest SR (88%) and lowest CR

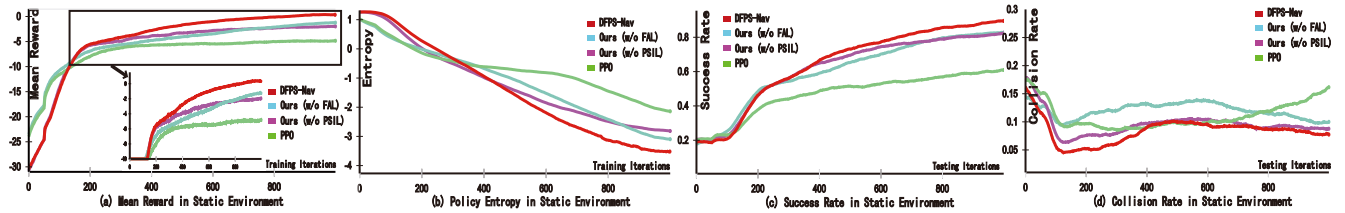

Fig. 4. Training and Testing Process Comparison in the Static Environment. The figure illustrates the performance of DFPS-Nav against the PPO baseline and its ablation variants on four metrics: (a) Mean Reward, (b) Policy Entropy, (c) Success Rate, and (d) Collision Rate. All curves demonstrate the significant advantages of DFPS-Nav in both convergence speed and final performance.

(7.5%), a segment length of 10 achieves the highest SPL (0.026) and SCT (0.082), despite its slightly lower SR (80%) and higher CR (11.5%). Therefore, considering the overall policy quality and efficiency, segment length 10 was chosen in our experiments.

Most importantly, as shown in Fig. 4 (a) and (c), DFPS-Nav achieves better performance in both mean reward and success rate compared to all other methods. This validates the core design of our framework: the PSIL module enables the agent to rapidly master effective behaviors by prioritizing the imitation of high-quality successful trajectories, thereby accelerating policy optimization. Concurrently, as seen in Fig. 4 (d), DFPS-Nav exhibits a remarkably low collision rate from the early stages of training. This is credited to the FAL module, which extracts fine-grained avoidance signals from failed experiences, allowing the agent to learn robust recovery and avoidance strategies instead of overly conservative ones. The change in policy entropy (Fig. 4 (b)) further supports this, PSIL rapidly lowers entropy by imitating superior trajectories; FAL encourages valuable exploration, preventing premature entropy convergence. They establish a healthy dynamic balance in policy entropy together, making the policy performance more stable and efficient.

TABLE V

ABLATION STUDY OF DFPS-NAV COMPONENTS IN STA. ENV.

	Static Env.			
Modules	SR↑	CR↓	SPL↑	SCT↑
PPO	57%	15.5%	**0.035**	0.081
Ours (w/o PSIL)	84%	10.5%	0.025	0.079
Ours (w/o FAL)	80%	11.5%	0.026	**0.082**
Ours (DFPS-Nav)	**92%**	**7.5%**	0.023	0.080

TABLE VI

ABLATION STUDY OF DFPS-NAV COMPONENTS IN DYN. ENV.

	Dynamic Env.			
Modules	SR↑	CR↓	SPL↑	SCT↑
PPO	53%	**12%**	**0.038**	**0.081**
Ours (w/o PSIL)	56.5%	**12%**	0.037	0.079
Ours (w/o FAL)	66%	13%	0.034	**0.081**
Ours (DFPS-Nav)	**80%**	14.5%	0.028	**0.081**

D. Real-Robot Experiments

For robust deployment in a real-world setting, we adopted a hierarchical, hybrid navigation framework. We use the

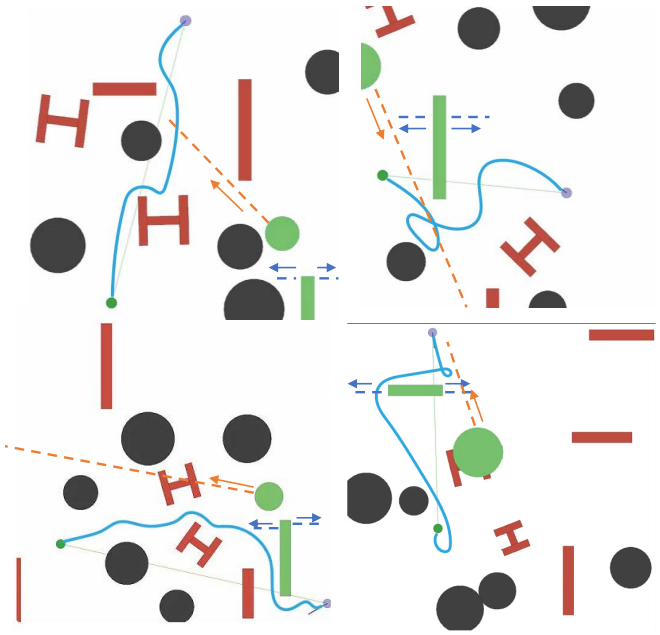

Fig. 5. Navigation trajectory in static and dynamic environments. The agent navigates from the starting point (light purple dot) to the goal (small green dot). The solid line represents the actual trajectory taken by the agent, while the dashed line shows the predicted or historical path of the dynamic obstacle (large green icon). Static obstacles are denoted by the black and red icons. Arrows along the paths indicate the direction of motion for both the agent and the dynamic obstacle.

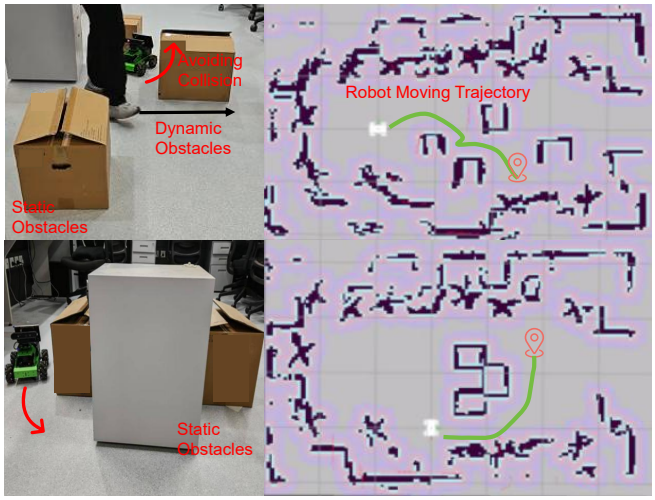

Fig. 6. The left image is an example of physical tests, and the right image is a visualization of a navigation trajectory in RViz.

well-established ROS2 Nav2 stack to generate a long-distance path through complex environments. To bridge the gap between simulation and reality, the sensory input was matched to the agent's observation space. The raw 360-point data was down-sampled to 180 points to conform to the observation space dimensionality that the policy was trained on. Our trained DFPS-Nav policy network serves as the local planner. It is responsible for real-time, reactive obstacle avoidance and precise local control, executing the global plan while handling unforeseen hazards.

A demonstration of this framework is illustrated in Fig. 6. As shown in Fig. 6 upper left, the robot initially follows the path provided by the global planner. When a dynamic obstacle crosses its path from the front, our DFPS-Nav policy, acting as the local planner, takes over. It executes a safe and reactive maneuver by first decelerating and then steering to the right to avoid a collision. After the obstacle has passed, the robot seamlessly readjusts its trajectory and proceeds to the goal, as shown in Fig. 6 upper right. This successful trial validates the policy's effectiveness in handling unforeseen dynamic events in the real world.

V. Conclusions

In this work, we presented DFPS-Nav, an asymmetric learning framework that dramatically enhances on-policy DRL for robot navigation by leveraging both successful and failed experiences. Our framework integrates Prioritized Self-Imitation Learning (PSIL), which acts as an integral guidance mechanism for trajectory-level quality-driven learning, with Failure-Aversion Learning (FAL), which employs a derivative-like analysis for segment-level credit assignment in failed attempts. This unique global-local learning paradigm significantly improves sample efficiency, achieves higher success rates, enhances path quality, and accelerates convergence in complex, dynamic environments. Despite these advancements, DFPS-Nav currently relies on engineered trajectory quality metrics and segment definitions. Future work will explore learning these evaluation criteria adaptively through meta-learning or self-supervised methods.

References

[1] J. Yang, Z. Meng, X. Xu, K. Chen, E. L. Li, and P. G. Zhao, "Task-oriented edge-assisted cooperative data compression, communications and computing for ugv-enhanced warehouse logistics," in *CCNC*. IEEE, 2025, pp. 1–8.

[2] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *CVPR*, 2019, pp. 6629–6638.

[3] M. S. Mondal, S. Ramasamy, J. D. Humann, J. M. Dotterweich, J.-P. F. Reddinger, M. A. Childers, and P. Bhounsule, "A robust uav-ugv collaborative framework for persistent surveillance in disaster management applications," in *ICUAS*. IEEE, 2024, pp. 1239–1246.

[4] K. Alcedo, P. U. Lima, and R. Alami, "Perspective-shifted neuro-symbolic world models: A framework for socially-aware robot navigation," *arXiv preprint arXiv:2503.20425*, 2025.

[5] L. Li, W. Zhao, C. Wang, A. Fotouhi, and X. Liu, "Nash double q-based multi-agent deep reinforcement learning for interactive merging strategy in mixed traffic," *Expert Systems with Applications*, vol. 237, p. 121458, 2024.

[6] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674–691, 2021.

[7] K. C. Zeng, S. Liu, Y. Shu, D. Wang, H. Li, Y. Dou, G. Wang, and Y. Yang, "All your {GPS} are belong to us: Towards stealthy manipulation of road navigation systems," in *USENIX*, 2018, pp. 1527–1544.

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

[9] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-imitation learning," in *ICML*. PMLR, 2018, pp. 3878–3887.

[10] Z. Chen and M. Lin, "Self-imitation learning for robot tasks with sparse and delayed rewards," in *ICMA*. IEEE, 2021, pp. 477–482.

[11] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, 1988.

[12] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *NeurIPS*, vol. 30, 2017.

[13] K. Xu, R. Chen, S. Zhao, Z. Li, H. Yu, C. Chen, Y. Wang, and R. Xiong, "Failure-aware policy learning for self-assessable robotics tasks," in *ICRA*. IEEE, 2023, pp. 9544–9550.

[14] Y. Guo, L. Xu, J. Liu, D. Ye, and S. Qiu, "Segment policy optimization: Effective segment-level credit assignment in rl for large language models," *arXiv preprint arXiv:2505.23564*, 2025.

[15] M. Bettini, A. Prorok, and V. Moens, "Benchmarl: Benchmarking multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 25, no. 217, pp. 1–10, 2024. [Online]. Available: http://jmlr.org/papers/v25/23-1612.html

[16] Y. Tang, C. Zhao, J. Wang, C. Zhang, Q. Sun, W. X. Zheng, W. Du, F. Qian, and J. Kurths, "Perception and navigation in autonomous systems in the era of learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 9604–9624, 2022.

[17] J. Escobar-Naranjo, G. Caiza, P. Ayala, E. Jordan, C. A. Garcia, and M. V. Garcia, "Autonomous navigation of robots: optimization with dqn," *Applied Sciences*, vol. 13, no. 12, p. 7202, 2023.

[18] H.-C. Jang, Y.-C. Huang, and H.-A. Chiu, "A study on the effectiveness of a2c and a3c reinforcement learning in parking space search in urban areas problem," in *ICTC*. IEEE, 2020, pp. 567–571.

[19] Z. Xu, X. Han, H. Shen, H. Jin, and K. Shimada, "Navrl: Learning safe flight in dynamic environments," *IEEE Robotics and Automation Letters*, 2025.

[20] D. C. Crowder, D. M. McKenzie, M. L. Trappett, and F. S. Chance, "Hindsight experience replay accelerates proximal policy optimization," *arXiv preprint arXiv:2410.22524*, 2024.

[21] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *ICLR*, 2016.

[22] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962–4971, 2022.

[23] Y. Zhou, S. Li, and J. Garcke, "Foresight social-aware reinforcement learning for robot navigation," in *CCDC*. IEEE, 2023, pp. 3501–3507.

[24] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[25] L. Li, W. Zhao, and C. Wang, "Pomdp motion planning algorithm based on multi-modal driving intention," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1777–1786, 2022.

[26] L. Chang, L. Shan, W. Zhang, and Y. Dai, "Hierarchical multi-robot navigation and formation in unknown environments via deep reinforcement learning and distributed optimization," *Robotics and Computer-Integrated Manufacturing*, vol. 83, p. 102570, 2023.

[27] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *ICLR*, 2016.

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

[29] D. A. Pomerleau, "ALVINN: an autonomous land vehicle in a neural network," in *NeurIPS*, 1988, pp. 305–313.

[30] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[31] N. Yokoyama, S. Ha, and D. Batra, "Success weighted by completion time: A dynamics-aware evaluation criteria for embodied navigation," in *IROS*. IEEE, 2021, pp. 1562–1569.