IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Traversability-Aware Legged Navigation by Learning from Real-World Visual Data

Hongbo Zhang, Zhongyu Li, Xuanqi Zeng, Laura Smith, Kyle Stachowicz, Dhruv Shah, Linzhu Yue, Zhitao Song, Weipeng Xia, Sergey Levine, Koushil Sreenath, Yun-hui Liu, *Fellow, IEEE*

***Abstract*—The enhanced mobility brought by legged locomotion empowers quadrupedal robots to navigate through complex and unstructured environments. However, optimizing agile locomotion while accounting for the varying energy costs of traversing different terrains remains an open challenge. Most previous work focuses on planning trajectories with traversability cost estimation based on human-labeled environmental features. This human-centric approach is insufficient because it does not account for the varying capabilities of the robot locomotion controllers over challenging terrains. To address this, we introduce a novel real-world learning pipeline that unifies offline demonstrations, online reinforcement learning, and multi-modal perception to achieve robust legged navigation. The framework employs multiple training stages to develop a planner that guides the robot in avoiding obstacles and hard-to-traverse terrains while reaching its goals. We first develop a novel traversability estimator in a robot-centric manner. The training of the navigation planner is directly performed in the real world using a sample efficient reinforcement learning method. With the proposed method, a quadrupedal robot learns to perform traversability-aware navigation through real-world interactions in diverse offroad and unstructured environments. Moreover, the robot demonstrates the ability to generalize the learned navigation skills to unseen scenarios.**

***Index Terms*—Legged robot navigation, traversability estimation, reinforcement learning, real-world training**

I. Introduction

WITH recent advancements in legged locomotion control [1–3], quadrupedal robots can now perform robust locomotion over various terrains, including challenging ones like stairs or slippery surfaces. When deploying these robots in outdoor environments, they need to identify whether a region is traversable by, for example, detecting obstacles and avoiding these areas accordingly. The question is whether we can step beyond binary traversability estimation to include the cost of traversing different terrains, thus preferring paths with lower traversal costs. For example, as shown in Fig. 1, we prefer the robot to walk on a concrete walkway instead of a muddy off-road area, while still avoiding obstacles. It could be more optimal for the robot in terms of better stability of the locomotion controller, energy efficiency, and better maintenance of the hardware, *i.e.*, having a low traversability cost.

Fig. 1. The proposed framework enables a quadrupedal robot to learn to avoid hard-to-traverse terrains (such as muddy areas covered by leaves) and obstacles from its own interactions through training directly in the real world. The robot learns to utilize its onboard color and depth vision sensors to identify challenging terrains and obstacles while navigating toward the goals, marked by the blue dashed circles. The inclusion of RGB images allows the robot to identify additional terrain textures that are difficult to perceive with depth images alone during the navigation.

However, estimating the traversability cost of terrain and planning to avoid high-cost regions for legged robots is a challenging problem. It requires the robot to interpret visual textures and corresponding physical properties, such as friction and elasticity, of different terrains. For example, when humans encounter a muddy road with high traversability costs, we first see the mud's texture and infer its difficulty based on *previous experience*. We highlight the importance of learning from previous experience as it's difficult to accurately estimate the terrain's actual cost until we traverse it ourselves. For legged robots to develop similar intelligence, they must address these multi-modal learning challenges, integrating data from various sources. To estimate terrain traversability cost, the robot needs to fuse color data (terrain texture), depth data (identifying non-traversable regions like obstacles), and proprioceptive feedback (implicitly encoding the physics) in real time and the real world. This problem is further complicated by the complex dynamics of quadrupedal robots, which require not only a robust locomotion controller itself but also consider the stability of such a controller over different terrains.

To address this and achieve traversability-aware quadrupedal navigation autonomy in the wild, in this work, we propose developing a multi-stage hierarchical reinforcement learning (RL) framework. This framework learns from both simulation and real-world data including different modalities. Our approach emphasizes the development of an online traversability estimator based on the value function of the RL-based

This work is supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government via the Hong Kong Centre for Logistics Robotics. (Corresponding author: Hongbo Zhang)

H. B. Zhang, Z. Y. Li, X. Q. Zeng, L. Z. Yue, Z. T. Song, W. P. Xia and Y.-H. Liu are with the Department of Mechanical and Automation Engineering at the Chinese University of Hong Kong, Shatin, Hong Kong. L. Smith, K. Stachowicz, S. Levine and K. Sreenath are with the University of California, Berkeley. D. Shah is with Princeton University. (e- mail: hbzhang@mae.cuhk.edu.hk, yhliu@cuhk.edu.hk)

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE I
COMPARISON OF TRAVERSABILITY-AWARE NAVIGATION METHODS

Study	System Input	Robot-Centric Traversability	Traversability from Whole Body Performance	Generalization	Off-road/Unstructured	Map-Free
Proposed (Ours)	**RGBD + proprioception**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Ewen et al. [4]	RGBD	No	No	**Yes**	**Yes**	No
Gan et al. [5]	RGBD + proprioception	**Yes**	No	**Yes**	**Yes**	No
Roth et al. [6]	Depth + semantic segmentation	No	No	**Yes**	Not Demonstrated	**Yes**
Yang et al. [7]	Depth	No	No	**Yes**	Not Demonstrated	**Yes**
Brandao et al. [8]	Pointclouds	**Yes**	No	**Yes**	Not Demonstrated	No
Anushri et al. [9]	Pointclouds + proprioception	**Yes**	No	**Yes**	**Yes**	No
Frey et al. [10]	RGB + proprioception	**Yes**	No	**Yes**	**Yes**	**Yes**
Oh et al. [11]	RGBD + proprioception	**Yes**	No	**Yes**	Not Demonstrated	No

low-level locomotion controller. The estimated traversability cost, which assesses control performance, is then used to train a high-level traversability-aware navigation planner that integrates both depth and color images. We also hypothesize that training the planner directly in the real world frees us from relying on accurate and photo-realistic simulators, and removes the need to deal with sim-to-real gaps.

Our major contributions can be summarized as follows: (1) We develop a traversability estimator based on the robot-centric experiences for legged robots. Unlike previous work, which primarily estimates traversability based on velocity tracking error or predefined human-provided labels, we leverage the value function obtained during the training of the low-level locomotion controller. Through extensive benchmark comparisons, we demonstrate that our approach achieves better performance and enables a more holistic traversability estimation from the robot's own experiences. (2) We propose a navigation planner that leverages multi-modal input, incorporating both color and depth visual data. Instead of relying solely on geometric information, as most existing works do, our approach utilizes RGB-D data to extract rich terrain features, leading to better navigation performance. (3) To overcome the challenges of simulating high-fidelity visual textures across various terrains, we propose an efficient learning framework based on real-world reinforcement learning. To enhance sampling efficiency, we improve an off-policy RL framework to leverage both online and offline data, significantly accelerating and enhancing training. (4) Extensive real-world experiments show that the robot learns to reach goals, avoid obstacles and minimize traversability cost under various challenging unstructured, off-road scenarios. We show that by directly training the policy in the wild, the robot learns to find a near-optimal trajectory in 15-20 minutes. (5) Through extensive real-world experiments, we also demonstrate the generalization ability of the trained policy in entirely unseen environments, with novel terrains and obstacles.

II. RELATED WORK

Most previous attempts to divide the traversability-aware navigation problem into two subproblems: traversability estimation and path planning. We categorize the related work accordingly and summarize the most relevant systems in Table I.

A. Traversability Estimation

Traversability estimation is a critical component in autonomous navigation, enabling robots to assess the navigability of their environment. Methods for traversability estimation can be broadly categorized into two approaches: environment centric and robot centric.

1) Environment Centric Approaches: These methods focus on analyzing environmental features to determine traversability, often relying on exteroceptive sensors like cameras and LiDAR. They assess terrain characteristics such as geometry, semantics, and physical properties to infer traversability. Some methods usually define traversability by empirically judging the cost from geometric features including meshes [8], point clouds [9], or depth images [7]. These geometric-based methods may be sufficient under indoor and structured scenarios, but they struggle with terrains that lack distinct geometric features, such as sandy, soft, or slippery surfaces [12]. Other approaches involve classifying terrain types through semantic segmentation of visual data [6, 13–15]. often mapping semantic features to traversability cost based on human-provided labels. Also, [4] introduces a Bayesian inference framework designed to estimate terrain properties crucial for robotic navigation utilizing data from RGB-D with semantic segmentation. However, the accuracy of the semantic-based methods depends on the quality and diversity of the training data. Misclassifications can occur in unfamiliar environments, especially under unstructured and offroad environments. Other work also allows legged robots to actively estimate and predict physical properties of unknown terrains through pre-defined probing motion [16, 17]. However, the space of physical properties relevant to traversability (e.g., stiffness, friction, deformability) is vast and often task-specific. In practice, it is infeasible to enumerate or model every possible physical parameter that might impact locomotion performance. These methods typically rely on human prior knowledge rather than the robot's own experiences [18]. A key limitation of environment-centric methods is that they do not account for the robot's locomotion controller or its own embodied experience. As a result, they may misjudge terrain difficulty, particularly in cases where the traversability is highly dependent on the robot's control strategy or physical limits.

2) Robot Centric Approaches: The robot centric traversability estimation methods emphasize the robot's interaction with the environment, leveraging proprioceptive feedback to assess traversability. They consider the robot's specific capabilities and responses to different terrains. The most straightforward approach is to consider control performance metrics such as tracking error of commands (e.g., velocity) [10] or IMU variance[19]. These quantitative metrics are used as labels for supervised learning of environment

representations. Other approaches extend this research by learning estimators to predict traversal time [20], selected environment physics parameters [21], or collision and fall indicators [22] from a history of the robot's proprioception feedback. [5] employs inverse reinforcement learning to approximate a traversability reward function. However, the accuracy of this estimation is highly contingent upon the quality and representativeness of the demonstration data. If the collected trajectories are suboptimal or fail to encompass the full spectrum of terrain conditions, the resulting reward function may not generalize effectively to novel or diverse environments. While these methods link terrain traversability to specific control performance metrics, they do not capture the whole-body performance (including velocity tracking, base stability, motion smoothness, energy efficiency, and more) of a legged robot's full-order dynamics driven by its controller. We address this by developing a traversability estimator from proprioception based on the value function of the RL-based locomotion controller. Our approach offers a more comprehensive, controller-aware, and data-efficient estimation method that directly reflects the robot's embodied interaction with the terrain and is well suited for real-world deployment.

B. Learning based Navigation

While numerous methods can tackle the planning problem for quadrupedal robots [23], we focus on learning-based methods that provide a planning policy directly from the robot's vision input.

1) Learning from Expert Trajectories: Some prior work opts to use supervised learning by imitating expert trajectories or demonstrations provided by humans, such as [24–26]. In these methods, the robot is trained to reason expert actions from vision-based observations along demonstrated trajectories. This approach enables the robot to acquire behaviors quickly by mimicking successful strategies. However, the traversability cost can differ significantly across different embodiments. If the robot only learns from demonstrations provided by others (*e.g.*, humans), it cannot update and optimize its planning strategy based on its own experiences: the planning policy is limited to offline data.

2) Reinforcement Learning based Approaches: Reinforcement learning offers an approach that allows the robot to update its planning strategy based on its interaction with the environment, *i.e.*, to learn from online data. Usually, RL requires extensive data through trial and error, therefore, some prior attempts leverage photo-realistic simulation to train the robot extensively and directly transfer it to the real world, primarily in indoor environments such as [27–30]. However, for outdoor environments where large sim-to-real gaps exist, due to factors like varying lighting conditions and terrain types [10], zero-shot transfer of a color-vision-based policy from sim to real is still challenging.

3) Learning in the Real World: Some other attempts try to train the planning policy directly with data collected in the real world, such as on racing cars with state-based data [31] or vision-based data [18]. For more complex robots, like quadrupedal robots, there have been attempts to train state-based control policies from scratch [32, 33]. Researchers in [5] incorporate real-world demonstration trajectories and employ inverse reinforcement learning to derive reward maps and plan trajectories based on real-world data. However, their method faces challenges due to the exponential increase in computation time caused by high-order states and large input dimensionality. Using RL to deploy real-world training for quadrupedal robots to extract vision features, which are in a much higher dimension, remains an open question. This requires a sampling-efficient yet capable online learning algorithm to learn from high-dimensional vision data. To tackle this and obtain a traversability-aware planning policy for real-world deployment, we enable the quadrupedal robot to take advantage of both online data (its own experience) and offline data, such as expert demonstrations and embeddings from a pretrained color image encoder.

III. Formulation of Traversability-Aware Legged Navigation and Its Framework

In this section, we formulate the problem of traversability-aware navigation using quadrupedal robots by RL and develop the framework to tackle this problem.

A. Problem Formulation

The navigation problem builds on the general robot navigation objective: learn to have a collision-free path towards the goal, but augmented with an extra task of minimizing the trajectory-wise traversability cost. To be specific, given a sequence of depth images $I^{\text{depth}}_{t:t-m}$, a sequence of color images $I^{\text{rgb}}_{t:t-n}$ at time t and a goal position $\mathbf{x}^d$, the planner π^p_{rgbd} should learn to generate velocity steering commands $\mathbf{a}^{\mathbf{p}}_{\mathbf{t}}$ for the locomotion controller π^c to avoid obstacles and reach goal point $\mathbf{x}^d$, while minimizing the trajectory-wise traversability cost (*i.e.*, traversability-aware). To be specific, the entire problem can be divided into three subtasks:

- **Goal Reaching**: The robot should reach the desired goal $\mathbf{x}^d$ in minimal time.
- **Obstacle Avoidance**: The robot should avoid static and dynamic obstacles while traveling to the goal. Obstacle information can be perceived by the robot's depth vision.
- **Traversability Cost Minimization**: The robot should be aware of the interplay between terrain traversability and locomotion capability, minimizing the traversability cost over trajectories. The terrain's traversability should depend on the robot's own interaction experiences instead of predefined labels, semantic classifications, or human-centric experiences.

B. Framework

The proposed traversability-aware hierarchical-legged navigation framework consists of two parts: (1) a low-level locomotion controller π^c for the quadrupedal robot, which can track varying velocity and turning commands over different terrains, and (2) a high-level planner π^p_{rgbd} that considers the navigation goal, surroundings, and terrain textures to specify control commands. As illustrated in Fig. 2, the training of this framework consists of four stages.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

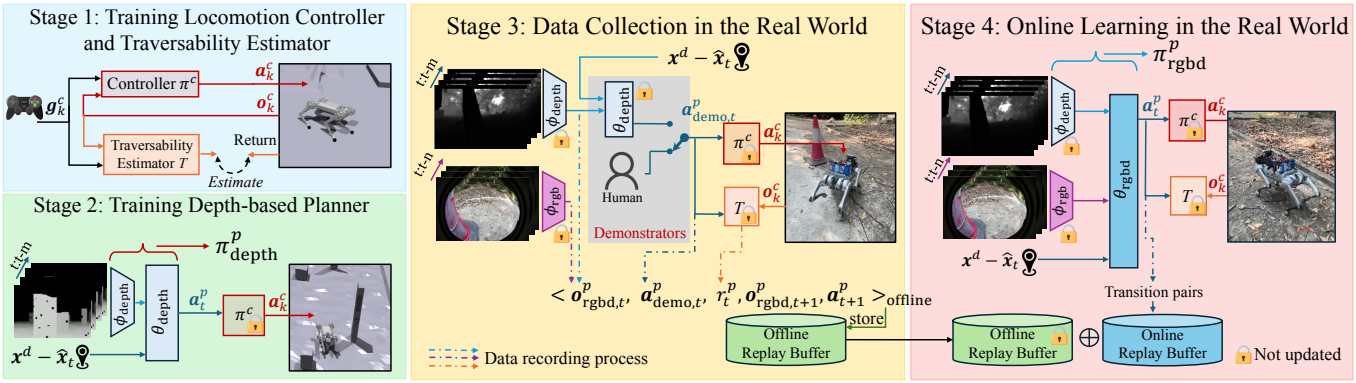


Fig. 2. The proposed training framework of high-level traversability-aware planner with RGBD input is divided into four stages. In Stage 1, the robust locomotion controller π^c tracking desired velocity commands $\mathbf{g}_k^c$ and the corresponding traversability estimator T are obtained as elaborated in Fig. 3. During Stage 2, a depth-based goal-reaching planner π^p_{depth} is trained using RL with access to the depth input. This planner will provide demonstrations in the next stage.In Stage 3, we collect real-world rollouts into datasets from both the depth-based planner π^p_{depth} and a human demonstrators. Data are collected in the form of transition samples. Both color and depth information are recorded. Actions $\mathbf{a}^p_{\text{demo},t}$ are recorded from demonstrators, and the online estimated traversability cost T is included as part of the reward r^p_t. Finally, during stage 4, the proposed planning policy π^p_{rgbd} is trained using both the samples from the offline dataset and the newly collected transition pairs using RLPD [34].

*1) **Stage 1**: Training a Robust Locomotion Controller and a Traversability Estimator:* Building the framework from scratch, we first obtain a robust locomotion controller π^c for the quadrupedal robot using RL, as shown in Fig. 3. Our novel design includes the development of an estimator for overall control performance changes induced by changes in terrain traversability T. This estimator is based on the critic network V^c obtained during the training of the control policy. Several design choices are implemented to ensure this estimator's functionality for traversability estimation on the real robot, detailed in Sec. IV-B. Both the controller and estimator are trained in simulation. The parameters of the locomotion controller and traversability estimator remain frozen during the subsequent training of the high-level planner.

*2) **Stage 2**: Training a Depth-Based Navigation Planner as a Demonstrator:* After obtaining the locomotion controller, we start training the planner π^p_{depth} on top of it. We aim to train a zero-shot deployable planner with only depth input in simulation, which will later act as a demonstrator during real-world data collection. The objective of this planner is to drive the robot to a target location while avoiding nearby obstacles, which are perceived through depth images. This planner will act as a demonstrator for autonomous data collection in the real world. Details will be introduced in Sec. V-A.

*3) **Stage 3**: Data Collection in the Real World:* To increase sample efficiency for the real-world training, we first collect rollouts in the real world with the trained depth-based planner π^p_{depth}. Additionally, our framework allows for the inclusion of human demonstrations, where human experts directly navigate the robot. During these rollouts, data is collected in the form of transition pairs: $\langle \mathbf{o}^p_{\text{rgbd},t}, \mathbf{a}^p_t, r^p_t, \mathbf{o}^p_{\text{rgbd},t+1}, \mathbf{a}^p_{t+1} \rangle$. These rollouts allow us to collect extended observations $\mathbf{o}^p_{\text{rgbd}}$, including RGB information that provides additional visual texture features of terrains and surroundings. The reward r^p_t collected during rollouts includes the estimated traversability cost T based on the robot's real experience interacting with the terrains. The details of this stage are discussed in Sec. V-B.

*4) **Stage 4**: Online Learning in the Real World:* After collecting the offline dataset, we can train a new planning policy π^p_{rgbd} from scratch by an off-policy RL method [35] using both the offline dataset $\mathcal{D}_{\text{offline}}$ and an online replay buffer $\mathcal{D}_{\text{online}}$, as illustrated in Fig. 2. Rollouts from the current policy π^p_{rgbd} are collected into the online replay buffer $\mathcal{D}_{\text{online}}$ and maintain the same format as those collected in the offline dataset. During training, the robot explores different trajectories, encounters varying traversability costs, and is encouraged to avoid high-cost regions by interpreting the cost through vision inputs of terrain textures. The details of training is introduced in Sec. V-C.

IV. Locomotion Control and a Traversability Estimator

In this section, we develop the first building block of the entire framework: the controller π^c for robust quadrupedal locomotion and a corresponding traversability estimator T as mentioned in the Stage 1 of the framework before.

A. Legged Locomotion Control over Complex Terrains

We first leverage RL to develop a robust blind locomotion controller for the quadrupedal robot, enabling it to traverse different terrains while tracking varying velocities. We use an actor-critic algorithm, Proximal Policy Optimization (PPO) [36], where the goal-conditioned actor policy π^c and critic network V^c are trained jointly in simulation. We adopt a symmetric actor critic structure to ensure that the critic can also be inferenced on the real-world robot. The controller is designed to track varying goal commands $\mathbf{g}_k$, including walking velocities in the sagittal and lateral directions $\dot{q}^d_{[x,y],k}$ and turning rate $\dot{q}^d_{\psi,k}$. The reward is designed to accomplish different control sub-objectives with different weights, including tracking given commands, following parameterized reference motion obtained by a central pattern generator [37], and several auxiliary terms including stabilizing base orientation,

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

minimizing base vertical velocity, joint velocity, and acceleration. We randomize robot dynamics and terrain features and the training is done in Isaac Gym [38].

B. Estimation of Control Performance Based on Value Function

Our design originates from the following intuition: if the robot can traverse challenging terrains easily, the estimated traversability cost will still be low. This indicates that the level of traversability should be scaled according to the strength of the locomotion controller. The critic network estimates the value, Interestingly, the traversability cost of the terrain can be inversely related to the estimated value from the critic network of locomotion representing the expected future accumulated reward.

However, directly using the critic network for real-world evaluation of traversability is challenging. A frequent problem we encountered is that the estimated value is also sensitive to changes in the given commands $\mathbf{g}_k$ using the goal-conditioned controller. For example, the critic return varies even when the robot locomotes on flat ground due to imperfect velocity tracking given different commands $\mathbf{g}_k$. This issue is further compounded by the sim-to-real gap: the critic may produce incorrect value predictions due to domain shifts, potentially overfitting to the simulation.

To address the issues above, we develop a traversability estimator T based on the critic network V^c trained with the actor-critic algorithm. Our major designs can be categorized into two key points:

- We design a flat ground baseline $V^c_{\text{flat}}(\mathbf{g}^c_k)$ and treat the value difference from the baseline to the critic value as the estimated cost. This aims to remove the dependency of the critic value on the given command $\mathbf{g}_k$.
- We explicitly calculate the uncertainty level σ of the estimation from critic T_{value} and use it to dynamically adjust the trust between the critic's estimation T_{value} and the estimation from a backup estimation T_{track}. This approach enhances the reliability of the estimation when the prediction from the critic is uncertain.

As explained in Algorithm 1, it contains three steps to obtain the final estimated traversability cost T to use in the real world, as detailed in the following.

Training Baseline Value Estimator: As illustrated in Fig. 3 (ii), we create a flat-ground value estimator $V^c_{\text{flat}}(\mathbf{g}^c_k)$ to capture the effects of commands $\mathbf{g}^c_k$ on performance. This estimator is trained by running the goal-conditioned locomotion controller π^c on flat ground, recording the ground truth return for various commands $\mathbf{g}^c_k$. The value estimator learns to predict the expected return from command input $\mathbf{g}^c_k$, providing a baseline to isolate the effects of terrain changes from command inputs.

Training Uncertainty-Aware Value Estimator: As shown in Fig. 3 (iii), we use the parameters of the pre-trained V^c and add a *dropout* layer right before its output layer, initializing a value estimation network V^c_{terrain}. For inference, we perform n forward passes with the same observation ($\mathbf{o}_k$) and condition ($\mathbf{g}_k$). We then calculate the mean of the predictions $\frac{1}{n}\sum_{i=1}^{n} V^{c,(i)}_{\text{terrain}}(\mathbf{o}_k|\mathbf{g}_k)$ as the estimated return, and the standard deviation $\text{Var}(V^{c,(i=1:n)}_{\text{terrain}}(\mathbf{o}_k|\mathbf{g}_k))$ as the uncertainty metric, inspired by [39]. This procedure acts like ensemble methods, where disagreement among multiple predictions indicates higher uncertainty, suggesting that the predicted return is less trustworthy and should be treated with caution.

Remark 1: The dropout layer is only added after the actor and critic networks are trained. The reason we do not concurrently train the critic with dropout is that the added dropout layer introduces a higher variance and fitting error, making the training for the actor network unstable.

Algorithm 1 Traversability Estimation

GIVEN Trained locomotion controller and its critic network π^c, V^c
Output T_{value} with uncertainty

// **Train** baseline value estimator V^c_{flat}
for $j = 1$: max_training_iteration
  for $k = 1$: batch_size
    Sample $\mathbf{g}^c_k = \dot{q}^d_{[x,y,\psi],k}$ under **flat ground**
    Observe $\mathbf{o}^c_k$
    Run $\pi^c(\mathbf{o}^c_k|\mathbf{g}^c_k)$ and collect future return R_k
  end for
  Update $V^c_{\text{flat}}(\mathbf{g}^c_k)$ to fit return R_k
end for

// **Train** uncertain-aware estimator V^c_{terrain}
Initialize Add a dropout layer next to the final layer of V^c and obtain initial V^c_{terrain}
for $j = 1$: max_training_iteration
  for $k = 1$: batch_size
    Sample $\mathbf{g}^c_k = \dot{q}^d_{[x,y,\psi],k}$ and **complex terrain types**
    Observe $\mathbf{o}^c_k$
    Run $\pi^c(\mathbf{o}^c_k|\mathbf{g}^c_k)$ and collect future return R_k
  end for
  Update $V^c_{\text{terrain}}(\mathbf{o}^c_k|\mathbf{g}^c_k)$ with *dropout* to fit return R_k
end for

// **Estimate** T_{value} with uncertainty
Observe $\mathbf{o}^c_k$, $\mathbf{g}^c_k$
for $i = 1 : n$ inference
  Run i-th forward pass on $V^{c,(i)}_{\text{terrain}}(\mathbf{o}^c_k|\mathbf{g}^c_k)$ with *dropout*
$T_{\text{value}} \leftarrow V^c_{\text{flat}}(\mathbf{g}^c_k) - \text{mean}(V^{c,(i=1:n)}_{\text{terrain}}(\mathbf{o}^c_k|\mathbf{g}^c_k))$
with **uncertainty** $\sigma(V^{c,(i=1:n)}_{\text{terrain}}(\mathbf{o}^c_k|\mathbf{g}^c_k))$
end for

Composition of Traversability Estimator: The difference between the two expected returns: $V^c_{\text{flat}}(\mathbf{g}^c_k) - V^c_{\text{terrain}}(\mathbf{o}^c_k|\mathbf{g}^c_k)$, indicates the changes in control performance induced by the changes in terrain, diminishing the effects brought by changes in conditions using the goal-conditioned controller. If the difference is smaller, it means the robot can traverse the current terrain with performance more similar to traversing flat ground using the same commands, thus the traversability cost T_{value} based on value estimation is lower, and vice versa. Furthermore, we will choose not to trust the estimation if the prediction is very unreliable, indicated by high prediction uncertainty $\text{Var}(V^{c,(i=1:n)}_{\text{terrain}}(\mathbf{o}_k|\mathbf{g}_k))$. In such cases, we use

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

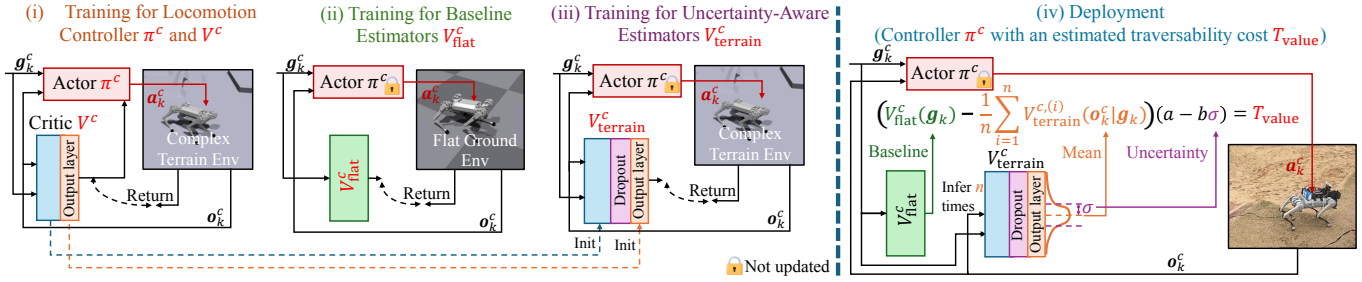


Fig. 3. The training framework for locomotion and traversability estimator. (i) The goal-conditioned controller (actor policy π^c) is trained to perform quadrupedal trotting gaits while tracking velocity commands $\mathbf{g}_k^c$ over various terrains. (ii) After obtaining π^c, a baseline value function V^c_{flat} is trained to estimate the return while the robot is walking on flat ground using the same actor policy. (iii) To create an uncertainty-aware estimator (V^c_{terrain}), we add a dropout layer before the output of V^c and fine-tune it on complex terrains. (iv) In real-world deployment, we perform n inferences using V^c_{terrain} for the same goal $\mathbf{g}_k$ and observation $\mathbf{o}_k^c$. The standard deviation σ of these inferences measures uncertainty. The mean value is adjusted by subtracting the baseline V^c_{flat} to reduce bias induced by the given goal $\mathbf{g}_k^c$. The negative of this difference is then multiplied by an adaptive weight $(a - b\sigma)$, where a and b are tunable parameters, resulting in the traversability estimation T_{value}.

another heuristic-based metric, such as command tracking errors, as an alternative for the traversability estimation. The composition of such a traversability estimation is given by:

$$T = w_1 T_{\text{value}} + (1 - w_1) T_{\text{track}} \tag{1a}$$

$$T_{\text{value}} = V^c_{\text{flat}}(\mathbf{g}_k^c) - \text{Mean}(V^{c,(i=1:n)}_{\text{terrain}}(\mathbf{o}_k^c | \mathbf{g}_k^c)) \tag{1b}$$

$$T_{\text{track}} = \|\dot{q}^d_{[x,y],k} - \hat{\dot{q}}_{[x,y],k}\|_2 \tag{1c}$$

$$w_1 = a - b\text{Var}(V^{c,(i=1:n)}_{\text{terrain}}(\mathbf{o}_k^c | \mathbf{g}_k^c)) \tag{1d}$$

where the weight w_1 is inversely proportional (by tuneable gains $a, b > 0$) to the uncertainty of the value estimation: lower uncertainty in the value estimation results in a higher weight for the value estimation T_{value} in the overall traversability estimation T, as we trust this estimation more. Conversely, higher uncertainty will lead the overall traversability estimation to rely more on heuristic-based metrics such as command tracking T_{track}.

In the next stage of training the navigation policy, the traversability estimator remains frozen and continues to provide real-time feedback, preserving a closed-loop interaction between high-level planning and low-level execution. Compared to fully training a joint locomotion-navigation policy, freezing the low-level modules ensures training stability, improves generalization, and significantly reduces training time, while still enabling responsive adaptation through real-time traversability inference.

Remark 2: This method of evaluating the difference between a baseline return estimation and an estimator for changes in the environment is not limited to traversability estimation as focused in this work. It can serve as a general evaluator of RL-based control performance. For example, it can also assess the sim-to-real gap: by comparing a baseline estimator for nominal dynamics parameters with an estimator for different dynamics parameters under the same given goal, a significant gap indicates a large sim-to-real discrepancy.

V. Combing Depth and Color Vision for Traversability-Aware Navigation

After obtaining low-level locomotion control π^c and a traversability cost estimator T that can be deployed on the real-world robot, we focus on the high-level goal-reaching planner in this section.

A. Depth-based Navigation Planner as a Demonstrator

As illustrated in Fig. 2, we first train a planner π^p_{depth} with access to a depth camera to navigate the robot to a given goal $\mathbf{x}^d$ in 2D Cartesian space while avoiding nearby obstacles. The trained planner will be zero-shot transferred to the real world and serve as a demonstrator in the next stage.

This planner is also trained using RL, and its Partially Observable Markov Decision Process (POMDP) design is detailed as follows.

1) Action: The depth-based planner π^p_{depth} outputs the change in control commands $\mathbf{a}^p_t = \mathbf{c}_t - \mathbf{c}_{t-1}$ with clamping to ensure smooth, bounded actions. This eliminates the need for reward tuning for smoothness. The planner is designed to operate at 5 Hz.

2) Observation: The observation of the planner $\mathbf{o}^p_{\text{depth},t}$ consists of: (1) a sequence of depth images $I^{\text{depth}}_{t:t-m}$, with a length of $m+1$; (2) the robot's current velocity $\hat{\dot{q}}_{[x,y,\psi],t}$ and the previous action $\mathbf{a}^p_{t-1}$; and (3) the position vector from the robot to the goal $\mathbf{x}^d - \hat{\mathbf{x}}_t$. This provides a short memory of the surroundings, an I/O feedback, and the direction towards the goal.

3) Reward: The reward for training this depth-based planner is defined as the weighted sum of three terms: r_{goal}, r_{FoV}, and $r_{\text{collision}}$, each defined as follows:

$$r_{\text{goal}} = \hat{\dot{q}}_{[x,y],t} \cdot (\mathbf{x}^d - \hat{\mathbf{x}}_t) + 10 G_{\text{arrive}} \tag{2a}$$

$$r_{\text{FoV}} = v_x \tag{2b}$$

$$r_{\text{collision}} = -G_{\text{collision}} \tag{2c}$$

The first reward term, r_{goal}, is the dot product of the robot's planar velocity $\hat{\dot{q}}_{[x,y],t}$ and the vector to the goal $\mathbf{x}^d - \hat{\mathbf{x}}_t$, encouraging a movement toward the goal. G_{arrive} provides a binary reward upon arrival. r_{FoV} discourages backward walking to maintain a forward-facing Field of View (FoV), while $r_{\text{collision}}$ penalizes the collisions.

4) Policy Representation and Training Details: The depth-based planner π^p_{depth} is a deep neural network with two components: a depth image encoder ϕ_{depth} and a base MLP.

We train the depth-based planner π^p_{depth} using PPO in Isaac Gym, where random obstacles (cylinders, boxes) and simulated noise are added to the depth images.

B. Data Collection in the Real World

To accelerate the subsequent online training in the real world, we choose to collect a real-world offline dataset with demonstration rollouts before the training. As illustrated in Fig. 2, the planner used during rollouts comes from (1) the depth-based planner π^p_{depth} trained in simulation and (2) human demonstrations where a human operator teleoperates the robot to navigate to the goal.

While the robot navigates by demonstrators, we collect transition samples from the planner over the trajectories. The transition sample at timestep t is denoted as $\langle \mathbf{o}^p_{\text{rgbd},t}, \mathbf{a}^p_t, r^p_t, \mathbf{o}^p_{\text{rgbd},t+1}, \mathbf{a}^p_{t+1} \rangle$, and will be stored in the offline replay buffer $\mathcal{D}_{\text{offline}}$. During deployment, a fisheye camera is used to capture wide-range color images, which are encoded using a pre-trained color image encoder ϕ_{rgb} trained on a large-scale dataset [40]. The details of the collected data are introduced as follows.

1) Action: The action $\mathbf{a}^p_t$ during the rollouts is still the change in the given control commands $\mathbf{a}^p_t = \mathbf{c}_t - \mathbf{c}_{t-1}$. Actions come from either the depth-based planner or human demonstrations.

2) Observation: The observation $\mathbf{o}^p_{\text{rgbd},t}$ at timestep t includes the latent embeddings from both the pre-trained depth image encoder ϕ_{depth} and color image encoder ϕ_{rgb}, the robot's estimated base velocity $\hat{\dot{q}}_{[x,y,\psi],t}$, the previous planner action $\mathbf{a}^p_{t-1}$, and the distance to the goal location $\mathbf{x}^d - \hat{\mathbf{x}}_t$.

3) Reward: To emphasize the traversability awareness, we add a new term r_{terrain}. The terrain reward is the negative of the estimated traversability cost T developed in Eq. (1). Also, we retain the goal-reaching term r_{goal} (2a) and forward term r_{FoV} (2b), drop the collision reward (as we lack a collision detector onboard).

$$r^p_t = 0.2 r_{\text{goal}} + 0.02 r_{\text{FoV}} + 0.5 r_{\text{terrain}} \tag{3a}$$

$$r_{\text{terrain}} = -T \tag{3b}$$

C. Online Learning for Traversability-Aware Navigation

After collecting the offline replay buffer, we begin training the final policy, the traversability-aware planner π^p_{rgbd} that fuses depth and color images, and robot state feedback. This training is from scratch and is directly deployed on the robot hardware while exploring different terrains during navigation.

We retain the depth image encoder ϕ_{depth} and color image encoder ϕ_{rgb} and online train the base MLP. This approach avoids learning directly from raw image data, which is of much higher dimensionality, thus accelerating real-world training. The training objective is to enable the base MLP to properly utilize the latent embeddings from the depth and color images, along with the robot feedback, to avoid terrains with high traversability cost and obstacles while navigating to the goal.

The elements in the POMDP (such as observation, reward, and action) during this stage remain the same as those in the offline data collection stage, as described in Sec. V-B. The key difference is that we now start updating the planner policy (which produces the action $\mathbf{a}^p_t$) using the data collected with the new policy. We adopt DroQ algorithm [41] (modified Soft Actor Critic (SAC) [35] with dropout in Q-value network), augmented it with RLPD. During each policy update, the sampled batch consists of two parts: half of the data is sampled from the offline dataset $\mathcal{D}_{\text{offline}}$ previously collected, and the other half is sampled from the online data $\mathcal{D}_{\text{online}}$ produced by the current policy. The update rule follows the standard SAC algorithm. By leveraging the offline dataset, we can accelerate online training and reduce the need for extensive exploration during online learning.

VI. Simulation Validation

In this section, we instantiate our proposed framework in a high-fidelity simulator to analyze various design choices and benchmark against prior work. This allows us to conduct large-scale experiments and derive statistically significant insights with scalable and reliable simulation evaluations, which will direct the real-world deployment of our system.

A. Simulation Setup

We developed a custom Gazebo simulation [42], as shown in Fig. 4, to study the various design choices. This simulation provides a consistent, controllable environment with physics simulation for the quadrupedal robot, including simulated depth and color ego-vision. In our setup, we assign **different friction coefficients** to different terrain types, such as lower friction in water pool regions and nominal values on grasslands, representative of real-world traversability. We also instantiate visually diverse obstacles, such as cones, trees, and even simulated humans, visualized in Fig. 4.

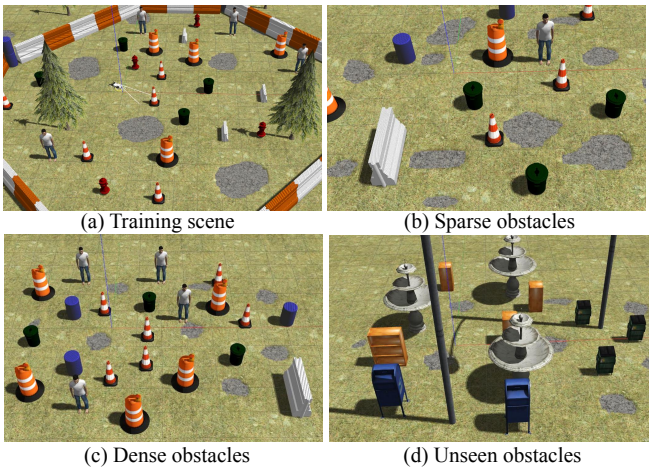
We deploy offline data collection and online training in Fig. 4(a) and test the trained framework in environments with *sparse obstacles* (Fig. 4(b)), *dense obstacles* (Fig. 4(c)), and *unseen obstacles* (Fig. 4(d)).

We report three metrics: average speed, traversability cost, and number of collisions, to quantitatively evaluate performance in tackling the navigation problem studied in this work, as described in Sec. III-A. The simulator provides access to ground-truth traversability costs by examining the average time the robot walks in the water pool area (with lower friction). In each environment, 20 navigation goal points are randomly sampled in the environment.

B. Estimating Traversability

As this work focuses on traversability-aware navigation, reliably estimating the traversability cost is one of our main objectives. In this benchmark, we compare different methods to estimate the traversability cost while keeping all other components in the framework the same:

- **Ours**: The proposed traversability estimation combines the value estimate T_{value} from the locomotion controller and a backup metric, the velocity tracking error T_{track}, as formulated in (1).
- T **from Tracking**: Only the velocity tracking error T_{track} is used to represent the traversability cost that the robot is currently traversing, as used in [10].
- T **from SysID**: Similar to [21], we adopt a teacher-student training paradigm. The student policy (resulting controller)



(a) Training scene (b) Sparse obstacles (c) Dense obstacles (d) Unseen obstacles

Fig. 4. The training scene (a) and three different testing scenes (b)(c)(d) in GAZEBO simulator are shown. To create environments with varying traversability, lower frictions are assigned to the areas of grey colors representing the water pool while nominal frictions are assigned to the green grasslands. Besides, various obstacles are randomly distributed inside. Policies are trained in the training scenario (a) and evaluated in testing scenarios (b), (c) and (d).

features an encoder that uses the robot's long I/O history to identify preselected environment parameters input to the teacher policy. Here, we choose foot friction as the parameter to estimate and use the identified friction to evaluate the traversability cost (in simulation, we only change the ground friction for different terrain types).

As shown in Fig. 5, our method, which includes the value estimate T_{value} to evaluate overall control performance, performs significantly better than using the velocity tracking error alone (T **from Tracking**), resulting in a lower total traversability cost over the trajectory (*i.e.* in *sparse obstacles*: 2.23 ± 0.52 s) compared with T **from Tracking** (3.14 ± 0.66 s), which uses the velocity tracking error alone. The results indicate that relying solely on heuristic-based metrics, such as velocity tracking error, fails to account for other factors that may not affect tracking performance but cause other degradations, such as foot slippage or body instability. These factors can be considered by the proposed estimator based on the controller's value function.

Furthermore, compared with T **from SysID**, which only identifies ground friction, our method also results in better trajectories with smaller accumulated traversability costs (ours 2.23 ± 0.52 s versus 3.44 ± 0.49 s). This improvement is due to the potential noise and unreliability in system identification. Additionally, there are many factors in real-world terrain that cannot be parameterized, such as ground filled with cobblestones or sand, where the system identification method can potentially fail to distinguish their difference using our proposed method.

C. Importance of Using Both Depth and Color Vision

We further highlight the importance of using both depth and color vision as input to the traversability-aware planner. In this benchmark, we develop different planners with different vision sources:

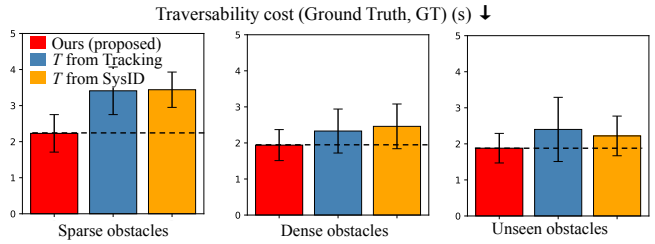


Fig. 5. Comparison between different choices of traversability estimation against ours is shown. The proposed traversability estimation method achieves at least **an 18% reduction** in traversability cost across the three testing scenes compared to the other two baseline methods. The evaluation is done on three testing scenarios with ground truth traversability cost represented by the time of the robot walking in water pool areas, obtained directly from the simulator.

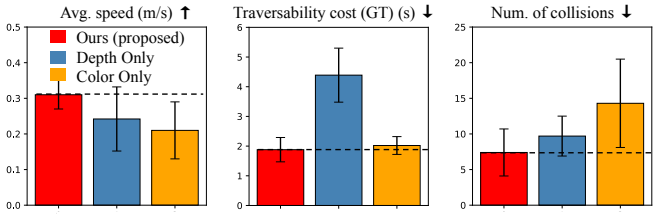


Fig. 6. Comparison between using single color or single depth vision as input against ours (using both color and depth) over the testing environment *unseen obstacles* (Fig. 4(d)) is shown.The proposed method shows significantly better performance, especially in reducing traversability cost, outperforming the depth-only approach by **133%**. Additionally, the proposed method reduces collisions with unseen obstacles during testing by **93%** compared to the RGB-only approach.

- **Ours**: The proposed method uses both depth vision and color vision, thus, our policy π^{p}_{rgbd} is based on RGB-D data.
- **Depth Only**: A planner that only has access to depth vision, using the same length of past depth images as the proposed method.
- **Color Only**: A planner that only has access to color vision, using the same length of past color images as the proposed method.

As shown in Fig. 6, the **Depth Only** method results in the highest traversability cost over the trajectory. This is expected, as terrain texture, which reflects traversability, cannot be easily distinguished by depth vision alone, which only provides geometric information.

When comparing our method, which combines both depth and color vision, with the **Color Only** method, the performance in terms of obstacle avoidance is similar in *sparse obstacles* (Fig. 4(b)) and *dense obstacles* (Fig. 4(c)), environments similar to the training environment. However, the performance of **Color Only** drops significantly in *unseen obstacles* (Fig. 4(d)), while our method still shows good performance. This comparison indicates that using only color images has a poor generalization ability over unseen obstacles. The inclusion of depth vision in our method alleviates this by providing geometric features.

D. Different Sources of Offline Data

We also note that the source of the offline data from which we collected also influenced the performance of the trained planner. In this benchmark, we validate the necessity of our

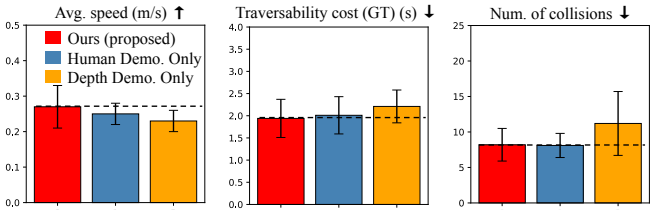


Fig. 7. Comparison between using a single data source of the offline data against ours (using demonstrations from both human experts and depth-based policy) is shown. The proposed method using the offline replay buffer containing the mixed demonstrations from both a human expert and a depth-based policy outperforms using a single data source of the offline dataset over environment *sparse obstacles* (Fig. 4(b)).

choice on generating the offline data from a mixture of human demonstration and the demonstration from the robot itself (the depth-based planner).

- **Ours**: The proposed method uses both the rollouts from trained depth-based planner π^p_{depth} and human demonstrations to generate offline data.
- **Human Demo. Only**: The rollouts are only collected from a human demonstrator to generate offline data.
- **Depth Demo. Only**: The rollouts are only collected from the trained depth-based planner π^p_{depth} to generate offline data.

As shown in Fig. 7, our method that uses the offline replay buffer which contains the mixed demonstrations from both a human expert and a pre-trained depth-based policy π^p_{depth} outperforms the choice of using a single data source of the offline dataset on the overall performances. (**Human Demo. Only** and **Depth Demo. Only**). This is because a mixture of sources of the offline data enhances the data diversity and therefore helps to avoid potential overfitting and does better exploration.

Compared with data collected solely from the depth-based planner (**Depth Demo. Only**), adding demonstrations from human experts provides a different distribution of the state-action pair to the offline dataset. This not only broadens the distribution of the collected trajectories but also introduces better trajectories in terms of minimizing traversability cost and obstacle avoidance, which are provided by human demonstrators.

However, if we only rely on the data collected by the human demonstrators, we also find there is performance degradation, especially for the goal-reaching objective (**Ours** 0.27 ± 0.06 m/s versus **Human Demo. Only** 0.23 ± 0.03 m/s). This is because human demonstrators may prefer the strategy that favors obstacle avoidance and lower traversability cost, but could be hard to consider minimizing traveling time. The data collected by the depth-based planner can overcome this problem as it explicitly handles the goal vector information with a well-trained planning policy.

E. Learning from Both Offline and Online Data

We compare our proposed learning method for the final planning policy with several benchmarks that share similar ideas of utilizing pre-collected offline data to benefit the training process.

- **Ours**: We use a standard SAC augmented with RLPD [34], which means that half of the update batch is sampled from the offline dataset $\mathcal{D}_{\text{offline}}$ and half of the update batch is sampled from the online dataset $\mathcal{D}_{\text{online}}$.
- **Vanilla SAC**: We train the planner policy using standard SAC [35] without the offline data.
- **Behavior Cloning (BC)**: We directly apply behavior cloning over the collected offline dataset $\mathcal{D}_{\text{offline}}$. A 2-norm action Mean Square Error (MSE) $||\mathbf{a}^p_{\text{demo}} - \pi^p_{\text{bc}}(\mathbf{o}^p_t)||_2$ is used as the loss function to minimize following the general behavior cloning setting, where π^p_{bc} is the target policy.
- **SAC with BC Loss**: Following [43], we use a standard SAC augmented with an extra behavior cloning loss term added to the actor loss.

Please note that all of offline data, excepting the one of the **BC**, is collected in the form of state-action transition pairs: $\langle \mathbf{o}^p_t, \mathbf{a}^p_{\text{demo},t}, r^p_t, \mathbf{o}^p_{t+1}, \mathbf{a}^p_{\text{demo},t+1} \rangle$, while we only collect state-action pairs $\langle \mathbf{o}^p_t, \mathbf{a}^p_{\text{demo},t} \rangle$ for **BC**.

As shown in Fig. 8, our method, which uses RLPD to incorporate offline datasets into the training, significantly outperforms the other baseline methods.

Compared to the **Vanilla SAC** method, which does not use any pre-collected dataset, we find that incorporating an offline replay buffer, as in our proposed method, significantly improves overall training performance. This advantage is particularly evident in our case, where we are addressing multi-modal learning from high-dimensional visual data. The pre-collected offline replay buffer, $\mathcal{D}_{\text{offline}}$, provides a high-quality dataset that helps to warm up the learning process. This not only enhances the sampling efficiency for online learning but also broadens the coverage of the distribution of collected trajectories, thereby improving the generalization of the learned planner.

We also compared our method with another approach that incorporates a pre-collected dataset during learning, but without using reinforcement learning: **Behavior Cloning (BC)**. As shown in Fig. 8, **BC** demonstrates worse performance because it relies solely on regression from state-action demonstration pairs without any reward labels. This method is more susceptible to suboptimal demonstrations and fails to explore behaviors that could be more optimal than those demonstrated. In contrast, our proposed method is less sensitive to the quality of the demonstrations because the transition pairs collected by our method include a reward term. After online exploration and learning, the agent optimizes for accumulated rewards and surpasses the performance of suboptimal demonstrations.

We also established a benchmark using a variation of behavior cloning, specifically **SAC with BC Loss**, as proposed in [43]. This method augments standard RL with a behavior cloning term, providing an alternative way to incorporate offline demonstration data into an RL training pipeline. While this approach outperforms other baselines, as shown in Fig. 8, our proposed method still delivers superior results. The reason is that the inclusion of a behavior cloning term during policy optimization can restrict the agent's ability to explore more optimal solutions based on its own experiences.

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

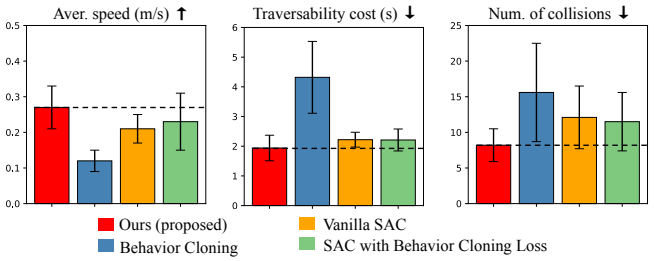


Fig. 8. Comparison of ways to incorporate offline dataset into the training process on testing scene *dense obstacles* (Fig. 4(c)) is shown. The proposed method, which combines offline and online data using RLPD, demonstrates at least **14%, 12%, and 40% better performance** across the three metrics. We validate that incorporating a pre-collected offline dataset is necessary concluding from the comparison to the benchmark **Vanilla SAC**. Then we compare with approaches that incorporate an offline dataset with supervised learning: **BC** and its variation **SAC with BC Loss**.

F. Summary

In summary, we justify several key design choices for training the RGB-D based planner policy and provide a comprehensive quantitative analysis. The findings are summarized in four key points:

- **Traversability Estimator**: We emphasize the use of value estimation T_{value} from the locomotion controller. This approach results in a more reliable and robust robot-centric traversability estimate.
- **Types of Vision Input**: We recommend using both color and depth information to enhance scene understanding and generalization of the outdoor environments, which benefits obstacle avoidance, terrain texture identification, and ultimately, traversability-aware navigation.
- **Sources of Offline Data**: We underscore the importance of collecting offline data from diverse sources. By incorporating both human-centric and robot-centric demonstrations, the dataset becomes more varied and beneficial for subsequent online learning.
- **Learning from Offline and Online Data**: We demonstrate that our proposed method, which integrates both offline and online data through reinforcement learning, improves sample efficiency and mitigates the effects of suboptimal demonstrations.

VII. Real World Experiments

After substantiating the design choices underlying our framework and demonstrating that our method achieves the best performance in simulation, we now deploy the entire system in the real world to validate its effectiveness.

A. Experiment Setup

The experiments are conducted on a quadrupedal robot (Unitree Go1). Visual sensing is achieved using a fisheye camera for color images and a RealSense D400 for depth perception. Additionally, a 3D LiDAR is utilized for localization via LiDAR-inertial odometry [44], providing the robot's relative position to the target point in outdoor environments.

To validate the proposed methods, we selected four distinct real-world scenarios, as illustrated in Fig. 9. These scenarios were chosen for their complex terrains, which include areas that are challenging to classify based solely on geometric information (*e.g.*, depth vision).

- **Roads with Leaves** (Fig. 9(a)): This scenario features terrain divided into difficult-to-traverse areas with piles of leaves and normal flat areas that are easier for quadrupeds to navigate. Construction cones are randomly scattered throughout as obstacles.
- **Cobblestone Roads** (Fig. 9(b)): The terrain in this scenario consists of a mix of cobblestone and flat road areas. The cobblestone sections cause slipperiness, leading to unstable movements for the legged robot, making these areas difficult to traverse. A fixed streetlight also serves as an obstacle.
- **Bunker Roads** (Fig. 9(c)): This scenario includes a sand-filled bunker in the middle, surrounded by flat grassy areas. The sand bunker is considered a difficult-to-traverse area as the quadruped often gets stuck. Construction cones are randomly placed as obstacles throughout the scenario.
- **Wilderness** (Fig. 9(d)): We deploy our robot in a genuinely wild and off-road forested environment that closely mirrors the unstructured, unpredictable conditions encountered in mountainous or remote, uninhabited regions. The terrain presents a diverse array of natural obstacles and traversal challenges, including widely scattered tree stumps, uneven rock formations, and dense underbrush, providing a realistic testbed for assessing the robot's locomotion capabilities in real-world field scenarios.

In each scenario, two or three goal points are selected, with arrival defined as the robot's position being within a 0.5-meter radius around the goal point. The goal points are switched consecutively after the robot successfully reaches each one, allowing for continuous data collection and training.

For real-world implementation, we first collect real interaction data and store them in the offline replay buffer $\mathcal{D}_{\text{offline}}$, collecting approximately 2500 sample points through demonstrations from both a human expert and a trained depth-based planner π^p_{depth}. This process typically takes around 10 minutes. Subsequently, online learning using the collected offline data by RLPD is conducted to train the final RGB-D based planner π^p_{rgbd}, as described in *Stage 4* (Sec. V-C).

B. Traversability Estimation in the Wild

As a key component of our navigation framework, we begin by presenting the quality and reliability of traversability estimation, offering both quantitative and qualitative analyses. We further compare our estimation method with semantic-based estimation methods to highlight its advantages. Additionally, we underscore the importance of explicit uncertainty evaluation for robust performance in real-world deployments.

1) Quality of the Traversability Estimation Result: We first visualize the explicit traversability estimation result under the four wild scenarios as shown in Fig 9. We visualize the estimated traversability cost collected during the training process by representing the level of traversability with varying shades of blue (darker blue indicates a higher traversability cost, meaning increased difficulty for the robot to traverse). We overlap the traversability estimation sample points onto the

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

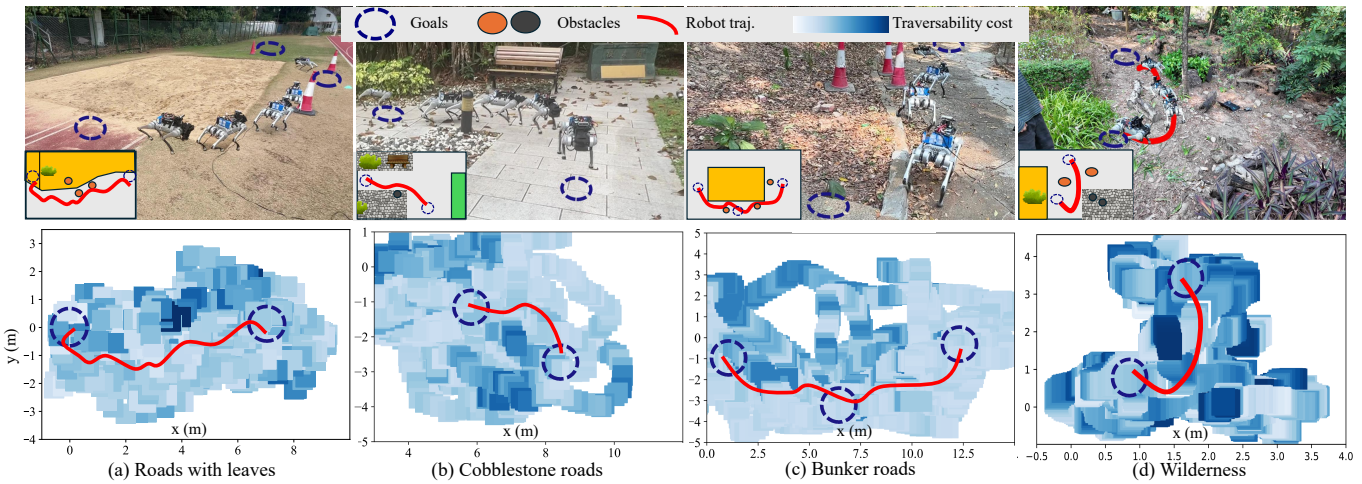


Fig. 9. Snapshots of the robot trajectories during the evaluation of the planner policy π^{p}_{rgbd} are shown. The robot successfully learns to avoid obstacles, avoid areas with high traversability cost, and reach goals (marked as dotted circles), with red curves representing the robot's trajectories. Simplified terrain schematics are provided to illustrate the top-down view of each environment setting. The second line of this figure showcases the traversability estimation results represented with deep (high traversability cost) and light(low traversability cost) blue shades using the data collected for the whole training process. The robot trajectories avoid most of the hard-to-traverse areas represented by darker blue shades as a desired behavior.

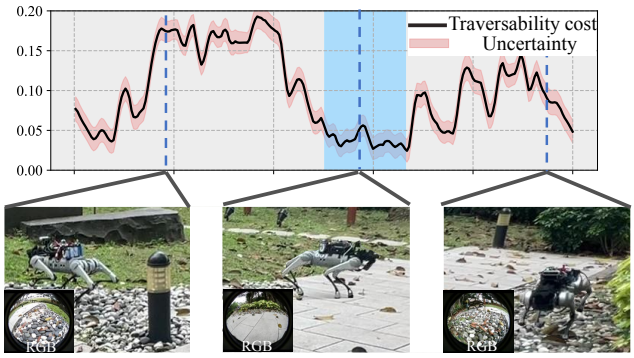


Fig. 10. Traversability cost estimation results (black curve) with uncertainty (red block) is shown in this figure when the robot is walking on a cross area of cobblestone roads and flat roads. During the time slots when the robot is walking on the cobblestone area, the estimated result has a high traversability cost, while on the flat roads, the estimated result shows a lower value close to zero (the blue area of the plot), which matches the real observation that the robot base shakes when slipperiness occurs between the foot and rocks when walking on the cobblestone area.

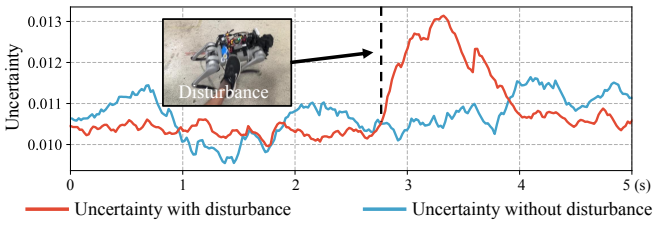


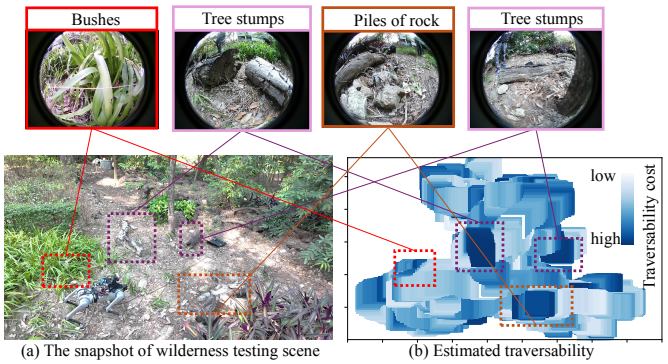
Fig. 11. The change of the uncertainty level of the traversability estimator when large disturbances that are not accurately simulated happen. The uncertainty level σ of the estimated traversability T_{value} represented by the orange line rises significantly after the disturbance occurs and then returns to a nominal level once the robot recovers from the disturbance. For comparison, we also plot the blue line representing the nominal walking situation on cobblestone terrain without disturbances, where the overall uncertainty level is lower.

ground truth position points from a top-down view. The areas with darker blue shades, representing high traversability cost, coincide with the actual difficult-to-traverse areas (such as piles of leaves, cobblestone regions, the sand-filled bunker, and several hard-to-traverse regions), demonstrating the reliability of the proposed traversability estimator. It is important to note that this traversability map is constructed solely for visualization purposes. Our approach does not rely on an explicit mapping between traversability and position, nor does it require a pre-built map labeled with estimated traversability during training.

Furthermore, to evaluate how effectively the estimator responds to changing terrain conditions, we visualize the estimated traversability values over a continuous time window as the robot moves across different types of terrain. As shown in Fig. 10, the estimation results accurately reflect changes in terrain. The final result of the scalar traversability cost T increases when the robot traverses cobblestone areas and decreases on nominal concrete roads, correctly reflecting the difficulty of traversing through these regions. For example, cobblestone areas are challenging for the robot to traverse and introduce obvious oscillation on the robot base.

2) Uncertainty-Aware Estimation: To emphasize our key design of explicit *uncertainty* evaluation for the estimation T_{value} described in Sec. IV-B, we examine the uncertainty level of the traversability estimator under scenarios that are not accurately simulated, *i.e.* a large disturbance. When this large disturbance happens, the estimation from V^{c}_{terrain} becomes uncertain about itself. As shown in Fig. 11, the uncertainty level σ represented by the orange line rises significantly after the disturbance occurs and then returns to a nominal level once the robot recovers from the disturbance. We also plot the level of uncertainty during nominal walking without disturbance when the uncertainty level is acceptably low and the estimation T_{value} is reliable, represented by blue lines. To have a reliable estimation, when the uncertainty level of T_{value} rises, less confidence is placed on T_{value} and the confidence on the backup estimation T_{track} from velocity tracking errors increases as described in Eq. (1).

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.



(a) The snapshot of wilderness testing scene (b) Estimated traversability

Fig. 12. Visualization of estimated traversability cost in the **Wilderness** with proposed method. (a) A third-person view of the scenarios. (b) The traversability estimation results from the robot's own experiences. The estimated cost values are recorded and projected onto a 2D coordinate map aligned with the robot's corresponding absolute positions. Several challenging, hard-to-traverse regions are highlighted along with their corresponding first and third person visual perspectives.

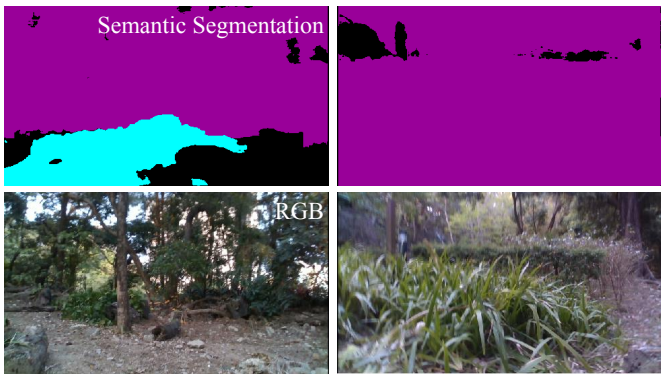


Fig. 13. Semantic segmentation [45] results (baseline) from the RGB image under unstructured environments. The segmentation lacks sufficient detail to provide the planner with the necessary information to distinguish between varying terrain textures like bushes, rocks, and tree stumps.)

3) Comparison against Semantic-based Estimation: We also compare our estimator with semantic-based methods [6], which assess traversability based on segmented semantic classes, assigning a fixed human-defined traversability cost to each class. Fig. 12 shows a more detailed result of the proposed estimation methods under *Wilderness* scene, where we highlight high-cost regions alongside the corresponding visual features captured by the robot's RGB fisheye camera. The results demonstrate that the estimator consistently identifies hard-to-traverse areas, such as dense bushes, tree stumps, and rock piles, aligning well with the visual cues from the environment.

In contrast, we also deploy the semantic-based estimation method from [6] as shown in Fig. 13 in the same scenario, *Wilderness*, and observe that the complex and mixed visual features of the unstructured terrain are difficult to segment accurately. As a result, the semantic based method tends to provide limited visual information and less reliable traversability estimation. Additionally, the mapping from segmented semantic classes to traversability usually relies on human prior knowledge or manual heuristics. Instead, our proposed method, which takes raw RGB images as input and infers traversability cost based on proprioceptive feedback, proves to be more suitable for off-road and unstructured environments.

C. Navigation Under In-distribution Scenarios

In this section, we evaluate the final navigation performance under in-distribution scenarios by deploying the trained policy in the same environment used during training and visualizing the robot's resulting trajectories.

- **Roads with Leaves** As shown in Fig. 9(a), the robot learns to avoid the leaf-covered region, which is challenging to traverse due to the piles of leaves and branches that can cause the robot's legs to get stuck and lead to instability. Instead of choosing the fastest route from the robot's starting point to the goal *i.e.*, a straight line across the leaf-covered region, the quadruped learns to identify the leaf-covered road as a high traversability cost area from visual input and avoids the region earlier before entering it. This is done by training with our proposed traversability-aware navigation framework. Additionally, the robot successfully learns to avoid obstacles and reach goals as specified by the general robot navigation tasks.
- **Cobblestone Roads** In the second scenario depicted in Fig. 9(b), the difficult-to-traverse area consists of cobblestones, which create slippery conditions and instability for the quadruped. Using our proposed method, the robot learns to identify this low traversability area from color input and chooses to navigate around the cobblestone areas to reach the goal, rather than directly crossing over them which could lead to a shorter but high traversability cost path.
- **Bunker Roads** In Fig. 9(c), we see a clear boundary between the bunker sand area and the flat area on the plot of the estimated high/low traversability cost. The robot learns to avoid such a sand-filled bunker. However, the robot does not adopt a conservative strategy where it leaves the boundary of the bunker too far. Instead, it walks just near the boundary. This is due to our training strategy, which integrates the robot's own experiences to learn from a combination of reward signals.
- **Wilderness** As shown in Fig. 9 (d), after training convergence, the robot successfully learns to navigate within the off-road environment. While progressing toward the goal, it identifies tree stumps, rock piles, and bushes from visual input and autonomously adjusts its trajectory to avoid these challenging regions. A more comprehensive trajectory visualization with more selected goal points is provided in Fig. 14.

The results demonstrate that the robot-centric traversability estimation from the locomotion system provides reliable signals for the planner to generate trajectories with lower traversability costs. The success of the real-world experiment validates that the robot learns to bind the color information with the level of traversability estimated from its own experiences, enabling it to avoid difficult-to-traverse regions based on visual input. The proposed framework enables the robot to

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

Fig. 14. We further evaluate the trained navigation policy in the in-distribution wilderness scenario by selecting multiple goal points. The blue circles indicate the chosen start and goal positions, while the red curves represent the robot's trajectories.

(a) Generalize to unseen environments with unseen obstacles: a suddenly appearing pedestrian

(b) Generalize to unseen environments with dynamic obstacles: a suddenly falling chair

(c) Generalize to unseen environments with visually similar terrain texture: a falling tree stump

Fig. 15. Testifying the generalization ability of the trained policy over several unseen scenarios.

navigate complex, unstructured environments in a robust and intelligent manner.

We also report performance metrics collected during the training process. As shown in Table II and Table III, goal-reaching and traversability-selection performance are measured in real time across multiple training timesteps: 1, 5, 8, 10, 15, 20, and 25 minutes. The results indicate performance improvements within the first 15 minutes across all scenarios. After this point, performance gains plateau and stabilize, suggesting that approximately 15 minutes of active training provides a favorable balance between training time and performance for these testing scenes.

TABLE II
AVERAGE GOAL REACHING TIMES (IN SECONDS) AT DIFFERENT TRAINING TIMESTEPS

	Average Goal Reaching Time (s)↓						
Scene	1min	5min	8min	10min	15min	20min	25min
Roads with leaves	102.6	64.2	37.1	**32.9**	Done	Done	Done
Cobblestone roads	79.4	58.1	41.1	28.3	**23.1**	Done	Done
Bunker roads	28.4	16.2	**15.4**	16.1	18.2	Done	Done
Wilderness	84.9	53.7	105.1	91.8	30.6	24.8	**19.0**

D. Navigation Under Out-of-Distribution Scenarios

To further validate the generalization ability of the trained policy π^{p}_{rgbd} with visual input in the real world, we evaluate the policy over two kinds of scenarios where the robot is not trained in: (1) Unseen environments with dynamic obstacles

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

TABLE III
AVERAGE TRAVERSABILTY RETURNS AT DIFFERENT TRAINING TIMESTEPS

	Average Traversability Returns ↑						
Scene	1min	5min	8min	10min	15min	20min	25min
Roads with leaves	-2.78	-2.38	-2.23	**-2.21**	Done	Done	Done
Cobblestone roads	-3.13	-3.17	-3.06	-2.89	**-2.54**	Done	Done
Bunker roads	-3.35	-3.04	-2.51	**-2.40**	-2.63	Done	Done
Wilderness	-3.75	-3.16	-3.22	-3.36	-3.13	-2.55	**-2.15**

(suddenly appearing pedestrian and suddenly falling chair) and (2) Unseen environments with unseen terrains.

1) Unseen Environments with Dynamic Obstacles: As shown in Fig. 15 (a)(b), we deploy the policy trained in the *Wilderness* scenario under a testing environment with entirely different surroundings, including red sidewalks and surrounding grass. A goal point is placed directly ahead of the robot.

- **Moving pedestrian** (Fig. 15(a)) During the movement of the robot with an average forward speed of $0.4m/s$ towards the goal guided by the policy, a pedestrian suddenly jumps out and blocks the way of the robot. We observe that the robot quickly stops moving forward and finds its path to avoid the suddenly appearing pedestrian.
- **Falling chair** (Fig. 15(b)) We also introduce a sudden disturbance by throwing a chair in front of the robot during its movement to evaluate its obstacle avoidance capability in rapidly changing environments. The robot promptly detects the unexpected obstacle and reacts accordingly to avoid it while continuing toward the goal.

These scenarios showcase the algorithm's ability to handle dynamic, real-time changes in the environment. We would like to emphasize that all of our testing scenarios involve environments that differ significantly from the training scenarios which only include static obstacles with standard geometric shapes and previously seen background.

2) Unseen Environments with Unseen Terrains:

- **Tree Stump** (Fig. 15 (c)) We placed a tree stump randomly on the road, which is undetectable by the depth camera but identifiable via the fisheye RGB camera. This navigation policy is trained in an entirely different environment (*Wilderness*, Fig. 9 (d)). In this new environment, the robot successfully avoided the stump using learned visual features, showing the ability to generalize beyond training scenarios. The robot's trajectory demonstrates its ability to recognize regions containing tree stumps as hard-to-traverse areas using RGB visual input from the forward-facing fisheye camera. It successfully avoids these regions and selects alternative paths with higher traversability. It is important to note that this is not a conventional obstacle avoidance case, as the low-lying tree stump cannot be detected by the depth camera. Instead, the avoidance behavior arises from the robot's learned ability to recognize and interpret terrain with similar visual features, gained through experience during training in the *Wilderness* scenario.
  This generalization test demonstrates that the trained policy effectively learns to extract key terrain features such as tree stumps from visual data during training. As a result, it can successfully recognize and avoid regions with similar terrain characteristics, even when deployed in entirely different environments.
- **Piles of Leaves** (Fig. 16) As shown in Fig. 16, unlike the training environment, where leaves are cluttered over a large area on one side of the concrete road, we randomly place small piles of leaves on the road. As demonstrated in Fig. 16, without any further fine-tuning, the robot is able to avoid the placed piles of leaves and reach the goal. This experiment highlights that the trained planner is not merely overfitting to a specific environment but has learned to generalize unseen environments, indicating its potential for application in more complex and large-scale outdoor scenarios.

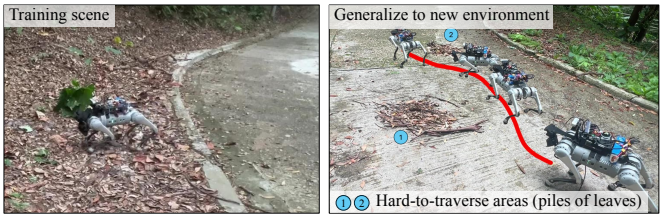


Fig. 16. The trained policy π^{p}_{rgbd} on Fig. 9(a) generalize successfully to the new environment over unseen terrains which is different from the training scenes. Two areas that are full of leaves marked with serial numbers 1 and 2 are set to represent the hard-to-traverse areas, which are unseen layouts for the planner π^{p}_{rgbd}.

E. System-level Comparison with Semantic-based Methods

We conduct a system-level comparison with a semantic-based traversability-aware method for navigation:

- **Proposed**: We train a navigation policy that directly takes color and depth images as input. The policy is optimized through online reinforcement learning, enabling it to learn traversability from the robot's own interactions and experiences within the environment.
- **Viplanner** [6]: This method trains a policy that takes as input a depth image, a semantic segmentation image, and the desired goal position, and estimates a coarse navigation plan through optimization. Traversability information is derived from both geometric data and semantic feedback provided by the sensors. We deploy this navigation system on our hardware platform and use the open-sourced policy checkpoint.

As shown in Fig. 17, the test is conducted in a wild environment where the goal point is located behind a pile of rocks. Both policies are able to guide the robot toward the goal; however, the proposed method generates a trajectory that actively avoids the rocky region, favoring safer, more traversable terrain. In contrast, the semantic-based policy leads the robot to move directly through the pile of rocks, reflecting limited awareness of terrain difficulty.

From the visualization of the semantic segmentation results, we observe that in such wild and unstructured environments, the semantic model tends to fail to accurately segment the rocky regions and cannot capture the detailed features of this complex terrain. As a result, the segmentation output lacks the

IEEE Transactions on Robotics (T-RO) paper, presented at ICRA 2026, Vienna, Austria. Cite as T-RO paper.

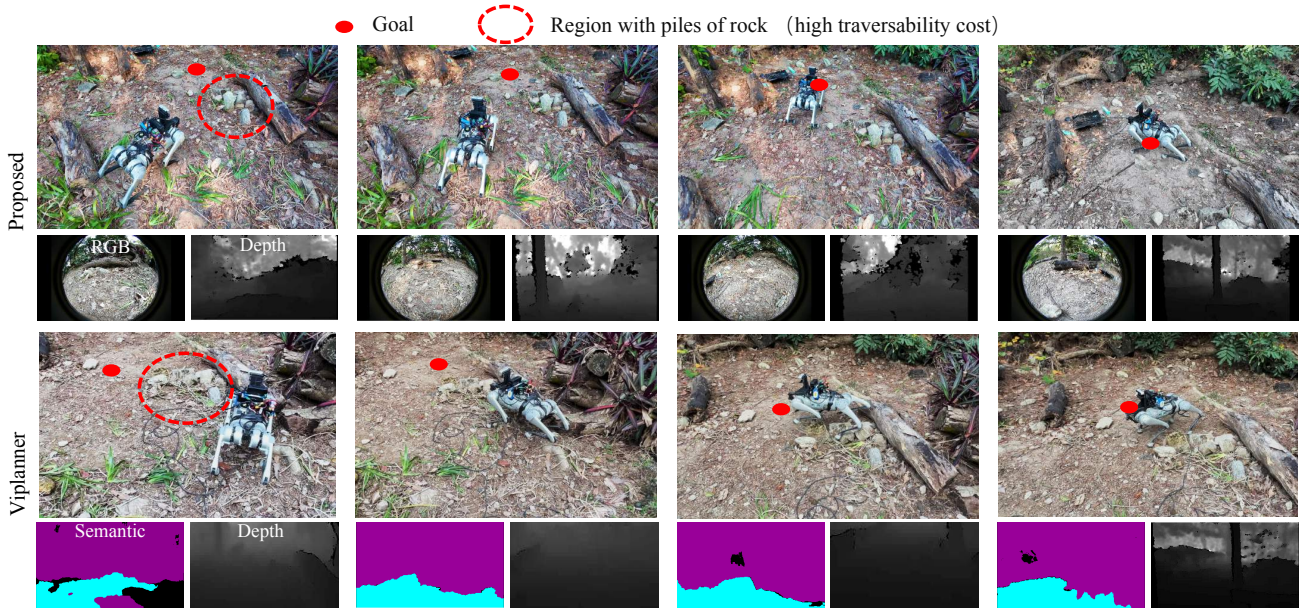


Fig. 17. Comparison between the proposed method and viplanner [6] under the same scenario. We deploy both the trained policies in a wild scenario, where the goal point is selected behind a pile of rocks. While both methods successfully guide the robot toward the goal, the proposed method generates a path that avoids the rocky region, demonstrating a preference for safer, more traversable terrain. In contrast, Viplanner directs the robot to move directly through the rock pile, indicating less sensitivity to terrain difficulty.

necessary resolution to distinguish between different terrain types, making it insufficient to support informed decision-making for the planner.

Moreover, our approach relying on the robot's own experiences to estimate traversability naturally assigns higher cost to regions containing rock piles through trials, enabling the planner to generate a path with traversability awareness. In contrast, semantic-based methods usually require each semantic class to be manually assigned a traversability level, typically based on human judgment, which may not accurately reflect the true difficulty of the terrain.

While the baseline method could potentially be improved by re-training with additional scene-specific or passively collected online data, such data lacks the interactive component crucial for learning meaningful terrain semantics. Our method, in contrast, collects online data through active interaction during deployment and leverages reinforcement learning to adaptively refine both terrain understanding and navigation strategy. This online interaction enables the planner to better perceive environment-specific traversability and make more informed decisions, especially in dynamic or previously unseen settings.

VIII. Conclusion and Future Work

In conclusion, we have introduced a multi-stage hierarchical reinforcement learning framework to achieve traversability-aware RGBD-based quadrupedal navigation. The framework is designed to train the navigation planner using real-world multi-modal data and the robot's own experiences interacting with various terrains. The key design choices within the proposed hierarchical framework are thoroughly validated through extensive benchmarks.

Our approach offers a novel way to train vision-based navigation planners that consider robot-centric traversability, making it easy to apply and customize planners for different robot platforms in complex outdoor environments. However, one limitation of our approach is that it struggles to generalize to entirely new environments with different terrain textures. For instance, a planner trained on *Roads with Leaves* might struggle with new terrain types, like a sand-filled bunker in *Bunker Roads*. To address this, a possible future direction could involve combining data collected from all different terrains into a unified offline dataset. Thanks to the flexible online traversability estimator, we can easily expand the offline dataset by collecting transition pairs as the robot navigates diverse terrains during routine operations. This extensive offline dataset can be used for zero-shot transfer with offline learning methods or adapted quickly to new scenarios using our proposed online learning approach with minimal rollouts.

References

[1] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning," in *Conference on Robot Learning*, 2023.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.

[3] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, "Perceptive locomotion through nonlinear model-predictive control," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023.

[4] P. Ewen, A. Li, Y. Chen, S. Hong, and R. Vasudevan, "These maps are made for walking: Real-time terrain property estimation for mobile robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7083–7090, 2022.

[5] L. Gan, J. W. Grizzle, R. M. Eustice, and M. Ghaffari, "Energy-based legged robots terrain traversability modeling via deep inverse reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 8807–8814, 2022.

[6] P. Roth, J. Nubert, F. Yang, M. Mittal, and M. Hutter, "Viplanner: Visual semantic imperative learning for local navigation," *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023.

[7] F. Yang, C. Wang, C. Cadena, and M. Hutter, "iplanner: Imperative path planning," *Robotics: Science and Systems (RSS)*, 2023.

[8] M. Brandao, O. B. Aladag, and I. Havoutis, "Gaitmesh: controller-aware navigation meshes for long-range legged locomotion planning in multi-layered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3596–3603, 2020.

[9] A. Dixit, D. D. Fan, K. Otsu, S. Dey, A. akbar Agha-mohammadi, and J. W. Burdick, "Step: Stochastic traversability evaluation and planning for risk-aware off-road navigation; results from the darpa subterranean challenge," *Robotics: Science and Systems (RSS)*, 2021.

[10] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," *Robotics: Science and Systems (RSS)*, 2023.

[11] M. Oh, B. Yu, I. Nahrendra, S. Jang, H. Lee, D. Lee, S. Lee, Y. Kim, M. K. Christiansen, H. Lim *et al.*, "Trip: Terrain traversability mapping with risk-aware prediction for enhanced online quadrupedal robot navigation," *arXiv preprint arXiv:2411.17134*, 2024.

[12] G. Haddeler, M. Y. Chuah, Y. You, J. Chan, A. H. Adiwahono, W. Y. Yau, and C.-M. Chew, "Traversability analysis with vision and terrain probing for safe legged robot navigation," *Frontiers in Robotics and AI*, vol. 9, p. 887910, 2022.

[13] F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt, "Geometric and visual terrain classification for autonomous mobile navigation," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2678–2684.

[14] D. M. Bradley, J. K. Chang, D. Silver, M. Powers, H. Herman, P. Rander, and A. Stentz, "Scene understanding for a high-mobility walking robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1144–1151.

[15] S. Ægidius, D. Hadjivelichkov, J. Jiao, J. Embley-Riches, and D. Kanoulas, "Watch your stepp: Semantic traversability estimation using pose projected features," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[16] H. Lin, H. Li, and Y. Gao, "See-touch-predict: Active exploration and online perception of terrain physics with legged robots," *IEEE Robotics and Automation Letters*, 2025.

[17] J. Chen, J. Frey, R. Zhou, T. Miki, G. Martius, and M. Hutter, "Identifying terrain physical parameters from vision-towards physical-parameter-aware locomotion and navigation," *IEEE Robotics and Automation Letters*, 2024.

[18] K. Stachowicz, D. Shah, A. Bhorkar, I. Kostrikov, and S. Levine, "Fastrlap: A system for learning high-speed driving via deep rl and autonomous practicing," in *Conference on Robot Learning*. PMLR, 2023, pp. 3100–3111.

[19] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti, "Learning ground traversability from simulations," *IEEE Robotics and Automation letters*, vol. 3, no. 3, pp. 1695–1702, 2018.

[20] G. B. Margolis, X. Fu, Y. Ji, and P. Agrawal, "Learning to see physical properties with active sensing motor policies," *Conference on Robot Learning*, 2023.

[21] A. Loquercio, A. Kumar, and J. Malik, "Learning visual locomotion with cross-modal supervision," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7295–7302.

[22] Z. Fu, A. Kumar, A. Agarwal, H. Qi, J. Malik, and D. Pathak, "Coupling vision and proprioception for navigation of legged robots," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 273–17 283.

[23] Y. D. Yasuda, L. E. G. Martins, and F. A. Cappabianco, "Autonomous visual navigation for mobile robots: A systematic literature review," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–34, 2020.

[24] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[25] A. Sadat, S. Casas, M. Ren, X. Wu, P. Dhawan, and R. Urtasun, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*. Springer, 2020, pp. 414–430.

[26] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "ViNT: A foundation model for visual navigation," in *7th Annual Conference on Robot Learning*, 2023.

[27] D. Jain, K. Caluwaerts, and A. Iscen, "From pixels to legs: Hierarchical learning of quadruped locomotion," in *Proceedings of the 2020 Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Kober, F. Ramos, and C. Tomlin, Eds., vol. 155. PMLR, 16–18 Nov 2021, pp. 91–102.

[28] M. Sorokin, J. Tan, C. K. Liu, and S. Ha, "Learning to navigate sidewalks in outdoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3906–3913, 2022.

[29] E. Wijmans, A. Kadian, A. S. 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*, 2019.

[30] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021.

[31] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2019.

[32] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.

[33] P. Wu, A. Escontrela, D. Hafner, K. Goldberg, and P. Abbeel, "Daydreamer: World models for physical robot learning," *Proceedings of Machine Learning Research*, 2022.

[34] P. J. Ball, L. Smith, I. Kostrikov, and S. Levine, "Efficient online reinforcement learning with offline data," in *International Conference on Machine Learning*. PMLR, 2023, pp. 1577–1594.

[35] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

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

[37] Y. Shao, Y. Jin, X. Liu, W. He, H. Wang, and W. Yang, "Learning free gait transition for quadruped robots via phase-guided controller," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1230–1237, 2022.

[38] J. Liang, V. Makoviychuk, A. Handa, N. Chentanez, M. Macklin, and D. Fox, "Gpu-accelerated robotic simulation for distributed reinforcement learning," in *Conference on Robot Learning*. PMLR, 2018, pp. 270–282.

[39] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[40] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine, "Rapid exploration for open-world navigation with latent goal models," in *Proceedings of the 5th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Faust, D. Hsu, and G. Neumann, Eds., vol. 164. PMLR, 08–11 Nov 2022, pp. 674–684.

[41] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka, "Dropout q-functions for doubly efficient reinforcement learning," in *International Conference on Learning Representations*, 2022.

[42] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, vol. 3. Ieee, 2004, pp. 2149–2154.

[43] J. Wu, Y. Zhou, H. Yang, Z. Huang, and C. Lv, "Human-guided reinforcement learning with sim-to-real transfer for autonomous navigation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.

[44] W. Xu and F. Zhang, "Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.

[45] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1290–1299.