2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

SCOML: Trajectory Planning Based on Self-Correcting Meta-Reinforcement Learning in Hybrid Terrain for Mobile Robot

Andong Yang, Wei Li and Yu Hu

***Abstract*— Trajectory planning is important for ground robots to achieve safe and efficient autonomous navigation in unstructured off-road environments. Most existing methods treat each terrain as a single type. However, in the real world, a ground usually consists of hybrid terrains. In this work, we propose a novel trajectory planning network that handles hybrid terrain. To further enhance safety, we have designed a self-correcting structure based on historical planning data. This structure can correct the trajectory when an inappropriate one is planned. To train the network, we introduce a two-stage training scheme based on Offline Meta-Reinforcement Learning, which can train the network with pre-collected non-optimal datasets and reduce the occurrence of hazardous planning. The proposed approach has been evaluated on both simulated datasets and a real robot platform. Compared to state-of-the-art baseline methods, the proposed approach reduces hazardous planning by 59.3% in hybrid terrains.**

I. INTRODUCTION

Autonomous navigation in unstructured off-road environments offers possibilities such as forestry monitoring, mining, planetary exploring, searching, and rescuing [1]. Trajectory planning is a key component of an autonomous navigation system [2]. In unstructured environments, trajectory planning involves considering both geometric and semantic information of the terrain to conduct safe and reasonable planning.

Several studies based on semantic segmentation have been proposed to extract semantic information of terrain in unstructured environments [3], [4], [5]. In those approaches, each pixel of the terrain image is assigned a unique label, which can represent specific terrain types such as grassland, gravel, or water. In other words, by default, each terrain type is assigned only one label. However, in reality, terrains often exhibit gradual changes, such as weeds growing on muddy ground or weeds growing on gravel surfaces. Hence, binary classification lacks evaluation of the underlay traversability cost. Also, These hybrid terrains are difficult to be directly segmented by semantic segmentation networks, and this distinction can have an impact on trajectory planning. As a result, such methods are weak in hybrid terrain, which is common in reality.

On the other hand, some methods directly employ end-to-end networks for trajectory planning without explicitly extracting terrain information. Such methods have more robust adaptability with different terrains. However, the accuracy and reliability of these methods can be limited due to restrictions in dataset quantity [6]. In addition, the end-to-end approach requires a lot of computing power, so it does not perform well in real-time planning [7].

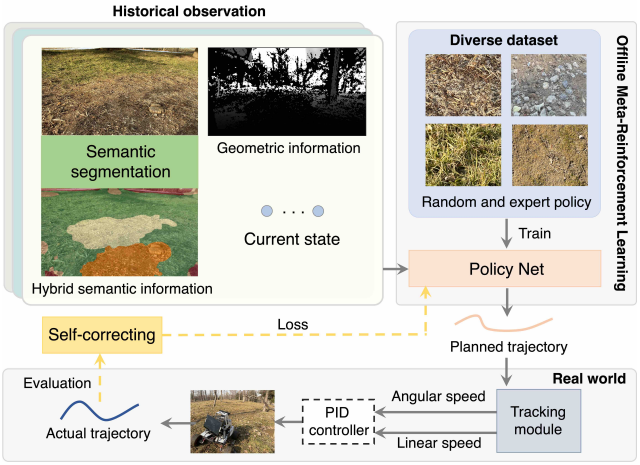

Fig. 1. This article proposes a trajectory planning framework for mobile robots in off-road environments that can handle hybrid terrain.

To tackle the problems mentioned above, in this paper, we first propose a terrain semantic information extraction method that can preserve hybrid terrain information to more accurately represent the semantic information of the terrain. More specifically, each terrain pixel can have multiple type labels. We then design a trajectory planning network that leverages the semantic and geometric features of the terrain to plan trajectories.

This design complicates the network inputs, making network training more challenging. Among existing methods, training networks typically require substantial interaction with the environment [8], [9], [10], which is difficult in unstructured environments and can result in unacceptable time costs. To alleviate this issue, we propose a training scheme based on offline meta-reinforcement learning. This method allows offline training of the network.

The trajectory tracking module is a subsequent module to the trajectory planning module, and it conducts specific commands for the robot to follow the planned trajectory. In off-road environments, the trajectory tracking module faces more significant challenges than in structured environments, affecting the robot's overall navigation performance [11]. To improve the rationality of the planning process, we introduce a self-correction loss. This loss can reflect the final driving

This work was supported by National Natural Science Foundation of China under Grant No. 62003323, No. 62176250 and Beijing Natural Science Foundation (L243008). Andong Yang, Wei Li and Yu Hu are with the Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China. {yangandong19b, liwei2019, huyu}@ict.ac.cn. Correspondence: Wei Li and Yu Hu.

performance of the robot when different trajectories are fed into the tracking module during network training. Therefore, the results generated by the trained network will be easier to execute by the tracking module. For instance, when excessively sharp turns are planned, or paths are generated through impassable terrains, feedback is provided to the network to correct such unreasonable planning.

In this paper, we propose SCOML, which extends the capability of trajectory planning methods for mobile robots in wild environments. The overall structure is shown in Fig. 1. To summarize, our contributions are (1) A new method to extract and represent semantic extraction of terrain, which can effectively reflect the hybrid terrain situations; (2) A training scheme based on offline meta-reinforcement learning is designed to address the challenge of collecting data in unstructured terrains for a real robot; (3) A self-correcting structure has been proposed, which can assess whether the planned trajectory is favorable for the subsequent tracking module to track, thereby enhancing the rationality of the planning process. In addition, the entire system is integrated into a real robot navigation system that includes sensing, mapping, planning, and control. We conducted experiments in the Gazebo simulator and in the real world to evaluate the effectiveness of the entire system.

II. RELATED WORK

Trajectory Planning. Trajectory planning for mobile robots in off-road environments is a highly nonlinear problem. Traditional methods often require a series of linearizations, which results in poor performance in unstructured environments [12]. Some learning-based methods have been proposed since they can effectively handle nonlinear problems. One popular approach is first to estimate the traversability of the terrain [13], [14] and then use methods like RRT* or A* for planning based on the estimated traversability [15]. The estimation of traversability is often derived from RGB images or LiDAR point clouds. However, these methods typically operate in SE(2) space and struggle to consider the geometric information of the terrain.

End-to-end methods are also one of the current popular directions [15]. Zhang et al. propose a method that utilizes deep reinforcement learning to directly establish a mapping from raw sensor data to planning [16]. Sathyamoorthy et al. use a self-supervised network to establish a mapping from the state of the robot and trajectory to rewards. It then selects the trajectory with the highest reward for execution [17]. These methods can consider both the geometric and semantic information of the terrain but require interaction with the environment, which can be challenging and risky in off-road environments. In addition, all the methods mentioned above assume that each terrain has only one type. However, in natural environments, hybrid terrains such as grasslands growing in muddy areas are more common. This poses a challenge for these methods in generating reasonable paths in such hybrid terrains.

Offline Meta-Reinforcement Learning. One of the simplest methods to utilize offline data is to pre-train a policy with imitation learning and fine-tune with on-policy RL [18]. However, offline data may be collected with different policies or tasks, resulting in sub-optimal training results. In addition, the training is inefficient since offline data is not utilized during finetuning. To this end, some methods try to combine meta-learning and offline reinforcement learning. Meta-learning assumes a distribution over environments (meta-tasks) and learns to adapt the policy to new environments that share similar structures [19], [20], [21]. This method can alleviate the issue of offline data being collected from different strategies from different tasks [22]. In this paper, different tasks are defined as trajectory planning in different terrain types. Previous offline Meta-RL studies have focused on the online setting, which means online training is still needed. Recent research has shown that if the offline dataset can be chosen appropriately, the online training can be removed without sacrificing too much performance [23]. Based on this research, we establish a planning policy network that uses terrain types to divide meta-tasks.

III. METHOD

The problem is to find the shortest trajectory from the current position to the end pose that satisfies smoothness and safety constraints. We use a two-dimensional uniform B-spline [24] to represent this trajectory. The parameters of trajectory are a set of control points $\boldsymbol{q} \in \mathbb{R}^{2\times M}$. The control points divide the path into M pieces, and the time duration of each piece is $\boldsymbol{T} \in \mathbb{R}^{M+1}$. At the time t, the three-dimensional point in the planned trajectory is $p(t) = \mathcal{B}_{\boldsymbol{q}}(t)$. In this work, we use a fixed time interval $\boldsymbol{T}$ to simplify the problem. The goal of this optimization is to find the $\boldsymbol{q^*}$ corresponding to the minimum cost.

As shown in Fig. 2, the proposed SCOML runs as follows. 1) Assign one or multiple type labels to each pixel of the terrain image. 2) Obtain the depth image represented in grayscale from the camera. 3) Select the current target based on the global path and the current robot position. Feed the results of steps 1) and 2), the robot's state and current target into the neural network for planning. 4) The tracking module will execute the resulting trajectory [11]. This procedure is repeated until the mobile robot reaches the goal.

A. The Design of SCOML

The SCOML first extracts the terrain semantic information and represents it by a value that indicates the level of difficulty for the robot to travel. This module first generates multi-label terrain types. This is adapted from Ga-nav [4], and the result is represented in the form of a mask $I^s \in \mathbb{R}^{w\times h}$, where h, w represent the height and width of the image. The values in I^s are determined based on the terrain type. Terrain types are classified into four categories: smooth terrain, rough terrain, bumpy terrain, and restricted terrain. Each terrain type is assigned a representative value to indicate the difficulty of traversal for that particular terrain [25]. The specific types and corresponding values are shown in Table I. Since directly obtaining the difficulty for robots to navigate in different terrains using existing methods is a

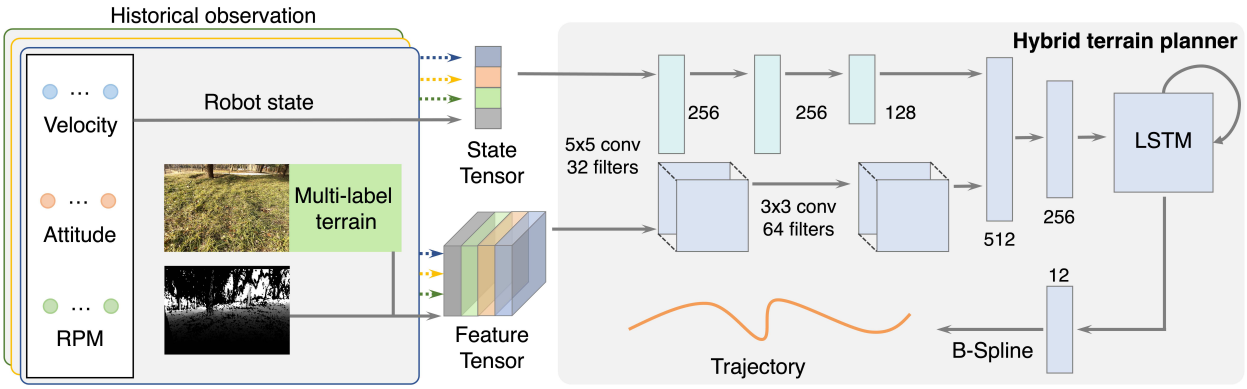

Fig. 2. The network structure of our method. The input data consists of three parts. The first part is the state of the mobile robot, including position, orientation, velocity, and RPM of each wheel. The second part is the terrain semantic information represented by multi-label terrain types. The third part is the terrain geometry information represented by a depth map. The hybrid terrain planner solves the final trajectory based on the prepared data. Then the tracking module converts the trajectory into specific control commands.

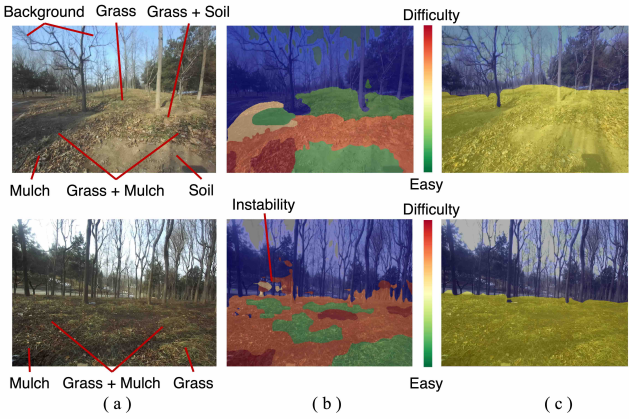

Fig. 3. Examples of the hybrid terrain types, with a redder color indicating greater difficulty in traversing. (a) The RGB image of the scene; (b) The hybrid terrain semantic information extracted by the method in this work; (c) The semantic information extracted by methods not explicitly designed for hybrid terrain.

challenge [26], this value is determined through experiments [27]. Set a fixed vortex-like path with different curvatures and measure the deviations of the robot as it tracks this path on various terrains. Determine the difficulty level of different terrains for this robotic system based on the deviation ratio. We use a piecewise function V to represent this table. The difficulty assessment value $M^s \in \mathbb{R}^{w \times h}$ based on terrain semantic information is obtained as follows. Let $c_i \in [0.0, 1.0]$ the output values corresponding to each type i from the softmax layer in the classification network. $M^s = V(argmax_i\ c_i)$, if $\exists c_i > 0.8$ and $M^s = \frac{1}{k}\sum_1^k V(c_k), c_k > 0.5$, if $\forall c_i \leq 0.8$. $i, k \in \{1, 2, 3, 4\}$. When none of the values for any type exceed 0.5, the corresponding value in M^s for that pixel is marked as 0, indicating that it is considered as restricted terrain due to a lack of estimation regarding the safety of the current terrain. For pixels that belong to the background, they are directly labeled as 0 because these pixels will not be traversed.

An example of terrain types and corresponding travel difficulty values is shown in Fig. 3. From Fig. 3 (b) and (c), it can be seen that the proposed method in this paper can extract more terrain semantic information compared to methods that are not specifically designed for hybrid terrain [4]. This enhances the rationality of subsequent planning. At the same time, we also note that this modification can lead to unstable classification results in some images. However, based on the experimental results Table II, the advantages brought by this modification are more significant despite the small range of instability.

The extracted terrain semantic information M^s will be fed into a neural network that is designed for planning. The network also has two other inputs. The first part is the state of the robot. At time t, the state is $\boldsymbol{s}_t$ which includes the position, orientation, velocity, and Revolutions Per Minute(RPM) of each wheel. The car chassis used in this work has four motors, with each motor independently driving a wheel. The RPM can help to determine if wheel slippage occurs. It also can provide real-time feedback on changes in the terrain. For example, when the vehicle encounters an obstacle, the RPM will suddenly decrease and then recover. Apart from terrain semantic information and the state of the robot, a depth map I^d is also fed into the neural network as input. I^d represents the geometry information of terrain and can easily obtained through a depth camera.

During robot navigation, historical data from the past time steps often contributes differently towards generating trajectory. For example, historical data can enable planning strategies to dynamically perceive the environment, gather information about inertia, and improve the capability to handle dynamic objects to some extent [28]. Thus, we introduce historical observations and inputs. Observations from a sequence of four past-time steps are stacked into a matrix. Along with the next target position s_g, the neural network input is denoted as $\boldsymbol{X} = [(s_{t-4}, M^s_{t-4}, I^d_{t-4}); ...; (s_t, M^s_t, I^d_t), s_g]$. When there are not enough historical observations at the beginning stage, we repeat the current observation as the historical observation.

All inputs are fed into the neural network $\boldsymbol{q} = \pi_\theta(\boldsymbol{X})$,

TABLE I

TERRAIN TYPES AND CORRESPONDING VALUES.

Terrain types	Pixel-wise labels	value
Smooth terrain	Concrete, Asphalt	1.0
Rough terrain	Soil, Grass	0.54
Bumpy terrain	Mulch, Rubble, Puddle, Mud	0.16
Restricted terrain	Trees, Logs, person, barrier	0.0
Background	Void, Sky	0.0

where θ is the parameter. The architecture of this network is shown in Fig. 2. The overall planning is established in a three-dimensional local coordinate system. The network's output $\boldsymbol{q}$ consists of 12 B-spline knots, which are used to generate the trajectory. In this paper, a fixed time interval is used to generate the trajectory. Based on this trajectory, the tracking controller controls the mobile robot to drive along the trajectory [11]. At time instance t, the inertial navigation system calculates an estimated pose of the mobile robot based on data from the Inertial Measurement Unit (IMU). Then, draw a circle with the mobile robot as the center and set the intersection point of the trajectory and the circle as the local target. If there are no intersections, the closest point on the trajectory to the robot, according to Euclidean distance, is selected as the next tracking point. This method allows the target to be maintained at a fixed distance, thereby alleviating the difficulty of tuning the tracking module. The target point is considered reached when the robot is within a certain distance from it. The general process of SCOML is provided in Algorithm 1.

B. Training Scheme

In existing methods, training such a network requires interaction with the environment using reinforcement learning methods, which can be inefficient. In this paper, to improve training efficiency and mitigate potential hazardous situations during interaction with the environment, a meta-reinforcement learning-based training scheme is proposed. This scheme allows for offline training of the network.

Consider a Markov Decision Process (MDP) associated with task $\mathcal{T}$ defined by $\mathcal{M}_{\mathcal{T}} = \{\boldsymbol{X}, \boldsymbol{q}, \mathcal{P}, r, \boldsymbol{X_0}, \gamma\}$, $\boldsymbol{X}$ is the state or network inputs, $\boldsymbol{q}$ is the action, $\mathcal{P}$ is the transition dynamics, r is the reward function, $\boldsymbol{X_0}$ is the initial state distribution, γ is the discount factor. In this work, the action $\boldsymbol{q}$ is the control point of the B-spline and is calculated by the policy network. In the offline meta-reinforcement learning setting, we divide the planning into N tasks based on terrain types. For each task $\mathcal{T}_{i=1,...,N}$, we collect data set $\mathcal{D}_{\tau_i} = [\boldsymbol{X}_1^i, \boldsymbol{X}_2^i, ...]$ based on policy $\pi_\theta^i(\boldsymbol{q}|\boldsymbol{X}, z)$, where θ is the parameter, z indicate different task. In this work, z is the different terrain types. The goal is to learn a meta-policy π_θ^* by optimizing the appropriate meta-training loss using the datasets.

$$\pi_\theta^* = \arg\max_\theta \sum_{t=0}^{T} [\gamma^t r(\boldsymbol{X}_t, \boldsymbol{q}_t)] \tag{1}$$

where T is time horizon which may be infinite.

Building upon [29], in this work, we do not directly maximize the reward but instead incorporate constraints during the training process to enhance its stability.

$$\pi_\theta^* = \arg\max_\theta \mathbb{E}_{\boldsymbol{X}\sim\mathcal{D}} \mathbb{E}_{\pi_\theta}[Q(\boldsymbol{X}, \boldsymbol{q})] \tag{2}$$
$$s.t \quad D_{KL}(\pi_\theta || \pi_\beta) \leq \epsilon$$

where $Q(\boldsymbol{X}, \boldsymbol{q}) = \sum_{i=t}^{T} \mathbb{E}_\pi[\gamma^{i-t} r(\boldsymbol{X}_i, \boldsymbol{q}_i) | \boldsymbol{X}_t, \boldsymbol{q}_t)]$ is the Q function, π_β corresponds to a mixture distribution over all past policies. Typically, π_β is estimated via maximum likelihood estimation, where samples from π_β are obtained simply by sampling uniformly from the data seen thus far: $\hat{\pi}_\beta = max_{\hat{\pi}_\beta} \mathbb{E}_{\boldsymbol{X}, \boldsymbol{q}\sim\pi_\beta}[log \hat{\pi}_\beta(\boldsymbol{q}|\boldsymbol{X})]$. This constrained optimization can be solved via Lagrangian duality:

$$\pi_\theta^* = \arg\max_\theta \mathbb{E}_{\boldsymbol{X}, \boldsymbol{q}\sim\beta}[log \pi_\theta(\boldsymbol{q}|\boldsymbol{X}) exp(A^\pi(\boldsymbol{X}, \boldsymbol{q}))] \tag{3}$$

where $A^\pi(\boldsymbol{X}, \boldsymbol{q}) = Q_\phi(\boldsymbol{X}, \boldsymbol{q}) - \mathbb{E}_{\boldsymbol{q}\sim\pi_\theta(\boldsymbol{q}|\boldsymbol{X})}[Q_\phi(\boldsymbol{X}, \boldsymbol{q})]$ is the advantage under current policy π. The critic Q_ϕ is trained to minimize the standard Bellman error [23].

The reward r used to train the network is defined based on the planned trajectory p and the actual state of the robot after executing the trajectory $\boldsymbol{X}^*$. We define the reward as:

$$r(p, \boldsymbol{X}^*) = \sum_{i=1}^{5} \alpha_i r_i(p, \boldsymbol{X}^*) \tag{4}$$

where α_i are normalized weights, $r_1(p, \boldsymbol{X}^*)$ is the success reward, a positive value will be given when the robot reaches the goal, $r_2(p, \boldsymbol{X}^*)$ is the length reward, which is trajectory length, $r_3(p, \boldsymbol{X}^*)$ is the smoothness reward, which is the sum of the changes in heading angle of the robot after driving along the trajectory, $r_4(p, \boldsymbol{X}^*)$ is the self-correcting reward. $r_5(p, \boldsymbol{X}^*)$ is the vibration reward, defined as the acceleration along the Z-axis. Maintaining lower vibration is beneficial for improving the quality and stability of sensor data.

Algorithm 1 SCOML

```
Initialize net π with trained parameters θ, Target sequence S_g, maximum iteration K.
Initialize current target position s_g.
while not reach the final target and K do
    Update current state s_t, I_t^d
    Update I_t^s based on function V
    Determine next local target s_g based on s_t
    if No history observation then
        Repeat current observation as history
    end if
    Stack observation to form input X
    Calculate knot q* = π_θ(X)
    Generate trajectory p(t) = B_{q*}(t), t ∈ T
    Record current observation
    Send trajectory to the tracking module
    Obtain control actions u_t and execute
end while
```

It has been demonstrated that adjusting the current planning based on historical data has a positive impact on

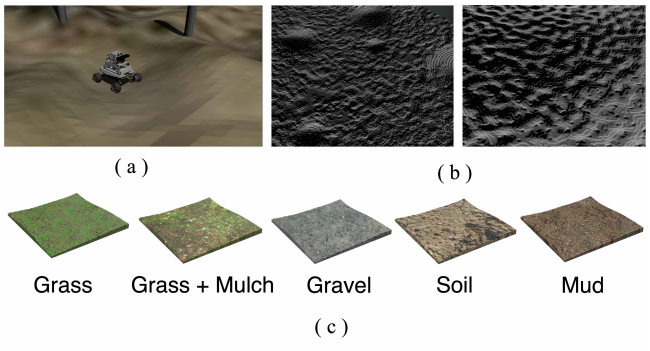

Fig. 4. (a) The 1:1 mobile robot model used in the simulator environment. (b) Examples of a randomly generated topographic map. (c) Examples of terrain textures used in the experiment.

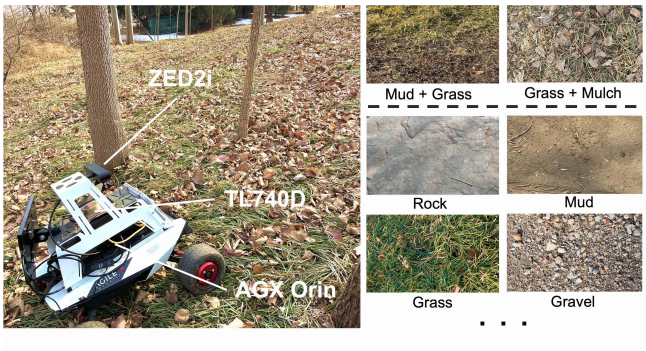

Fig. 5. The mobile robot used for experiments and part of terrains involved in experiments. Terrain in off-road environments typically exhibits various types and it can vary from single-type to hybrid terrain. In addition, off-road terrains also exhibit a wide range of geometric features.

trajectory planning in off-road environments [30]. Based on this, we have designed the self-correcting reward. The actual trajectory of the robot is denoted as p', and p is the planned trajectory. then the $r_4(p, \boldsymbol{X}^*) = ||p' - p||_2$ is the negative L2 distance between p' and p. This loss is set to 0 at the beginning since there is no historical trajectory available. The self-correcting reward can force the policy to plan trajectories that are easier for the tracking module to follow, thus avoiding scenarios with large angle turns, continuous acceleration, and deceleration to some extent.

In many cases, this policy will already perform the task successfully. However, if there is a significant difference between the test terrain and the terrain on which the training data was collected, the network may fail to plan a trajectory. In such cases, finetuning can be performed. Overall, this process does not require a large amount of data and can be completed with approximately 6 minutes of demonstration data. This is because the policy π^*_θ already provides a good initialization. The finetuning process is the same as the training process used in the offline training phase but utilizes online data collection.

C. Trajectory Tracking

The tracking module controls the robot to follow a certain trajectory. The result of planning is a continuous trajectory represented by a B-spline, and we obtain a series of tracking module targets through sampling. At time instance t, the inertial navigation system calculates an estimated pose of the mobile robot based on data from the IMU. The tracking module selects a position on the planned trajectory in fixed distance L from the current position as the next target point. If there is no point on the trajectory in L distance away from the current position, the tracking module selects the closest point on the trajectory to the current position as the target.

IV. EXPERIMENT

A. Hardware Setup

1) Real-World Setup: Our experimental mobile robot is a modified electric scout (weight 26 kg; LWH 0.612m × 0.58m × 0.295m), as shown in Fig. 5. The chassis was purchased from AgileX. Sensors equipped on the robot include a stereo camera ZED2i and an IMU RION TL740D. The onboard computer is an NVIDIA AGX Orin. All sensors and algorithms run under the Robot Operating System (ROS), and the planning frequency is set to 10Hz.

2) Simulation Setup: After training with the real-world collected dataset, we first test the training results using the GAZEBO [31] simulator to quickly adjust hyperparameters and evaluate the results. The terrain generation process in the simulator is as follows: First, a height map with undulations is obtained based on Gaussian sampling. Then, cylindrical and rectangular obstacles are sampled and randomly placed on the map. Finally, the height map is converted into terrain in Gazebo with different textures [32]. As shown in Fig. 4, different textures are applied to terrains. Friction between the terrain and the robot is adjusted to simulate the effects of various terrains on the robot. The terrain types will be directly given to the planning algorithm without using a separate network for classification. The mobile robot model used in GAZEBO is modeled 1:1 according to the mobile robot in the real world. It provides RGB-D camera frames and IMU data. The simulator runs on a desktop computer with an Intel Xeon E5-2650 v4 processor and Nvidia 2080Ti.

B. Data Collection

For offline meta-reinforcement learning, offline datasets consist of trajectories of states, actions, and associated rewards. This data can come from demonstrations, sub-optimal policies, or even just random exploration in the environment [33]. The offline dataset used for training in this paper is collected from single types of terrain and hybrid terrains, including grass, mud, soil, gravel, Mulch (leaves) mixed with grass, and gravel mixed with mud. Since collecting data from experts in the real world is significantly more time-consuming, we adopted a mixed data collection strategy. Following the categorization of meta-learning, we collected data separately for each type of terrain. The dataset for each terrain consists of three parts: 1) three demonstrations from an expert policy with each demonstration containing a 200 meters trajectory, 2) 10 suboptimal trajectories generated from a behavioral clone of expert policy with the same length

TABLE II

BASELINE COMPARISONS AND ABLATION STUDY.

Method	Baseline				Our			
	Traditional	**RAORN**	**Terp**	**BADGR**	**No history**	**No finetuning**	**No hybrid**	**SCOML**
Succcess Rate (%)	46±8	56±4	64±8	70±4	76±2	74±4	66±4	**78±4**
Smoothness (rad)	**0.58±0.11**	0.84±0.13	0.76±0.09	0.8±0.15	0.74±0.08	0.78±0.16	0.82±0.12	0.7±0.05
Norm. Traj. Length (-)	1.9±0.1	2.1±0.6	2.3±0.8	2.2±0.4	2.1±0.4	1.8±0.5	2.0±0.5	**1.7±0.2**
Hazard (-)	32±12	23±10	26±13	21±8	16±8	15±10	22±11	**13±7**
Consistency (m)	10.2±5.3	16.5±6.3	11.8±4.7	9.7±3.8	11.2±4.2	7.7±4.5	9.1±5.1	**6.9±3.2**

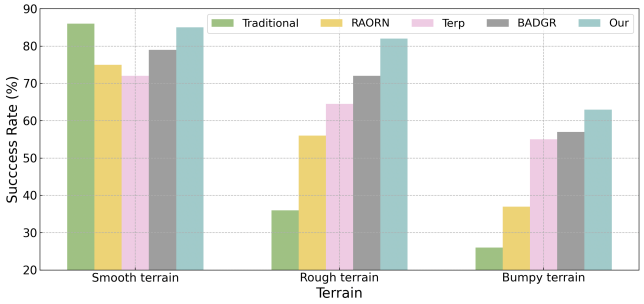

Fig. 6. Comparison of the success rates of different methods on different types of terrain.

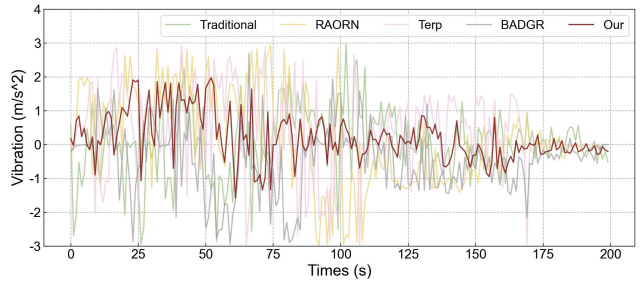

Fig. 7. Comparison of the vibration of different methods on a path that includes smooth, rough, and bumpy terrain.

[29], and 3) 2000 data pairs from random action sampling. On average, we obtain 5000 episodes of interaction data for each of the type of terrain.

All RGB images and depth images are resized with a resolution of 480 × 270 and a max range of 15 meters at 15 FPS. Each input $\boldsymbol{X}$ used for the neural network includes four historical observations and the current observation. We use software time synchronization to achieve synchronization between IMU data and image data. The control points of the collected trajectory are obtained through fitting.

C. Comparing with the State-of-the-Art

We compared our method with traditional approaches and learning-based methods. First, a traditional method based on an optimization is selected as a comparison [34] to represent the methods that do not consider terrain. This method performs planning based on an estimated traversability map and only considers the geometric changes in the terrain. Terp [35] is a learning-based planning method. This method first employs a fully-trained Deep Reinforcement Learning (DRL) network to generate a cost map for the current terrain, then generates locally optimal waypoints on the cost map and computes the feasible trajectory using another DRL-based method. This method considers both semantic and geometric information about the terrain but assumes that each terrain has only one type. Another learning-based method [36], we abbreviate this method as RAORN for simplicity, utilizes only one neural network to directly estimate the score for every trajectory based on classification information of RGB images and then selects the trajectory with the best reward. Similarly, this method assumes that each terrain has only one type. BADGR is an end-to-end learning-based navigation system [9]. It uses reinforcement learning to map sensor data directly to control commands. The dataset used to train the network contains hybrid terrains. Therefore, this method has a certain capability to handle hybrid terrains. However, due to the lack of direct handling of mixed terrains and the different distribution of planning strategies across different terrains, using a single network to fit all terrains may not yield satisfactory results.

To validate the effectiveness of our method, we conducted experiments on a 300 meters path that includes terrains present in the dataset, such as grass, gravel, and mud, as well as a hybrid terrain not present in the dataset, like mud with mulch, gravel with grass, roots and twigs terrain. The ratio of smooth, rough, and bumpy terrain is 1:2:1, and the ratio of terrains present in the training set to those not present in the training set is 4:1. To reduce the bias in the statistics, we consider successful planning when the robot travels 60 meters. The following metrics are used to quantify the performance of our method compared to other methods.

- **Normalized Trajectory Length**: Defined as the ratio of the actual trajectory length to the straight-line distance from the starting position to the target point.
- **Hazard**: The number of hazard planning occurrences during the path. Hazard planning is defined as: slide, collision, acceleration or speed exceeds the specified value, roll angle, or pitch angle of mobile robot exceeds 30 degrees, which are the upper limit of the robot's capabilities.
- **Smoothness**: The sum of the heading angle changes as a robot travels along a trajectory. The smaller value indicates the planned trajectory is smoother. In order to compare different methods on the same scale, we normalize this metric with the path lengths.

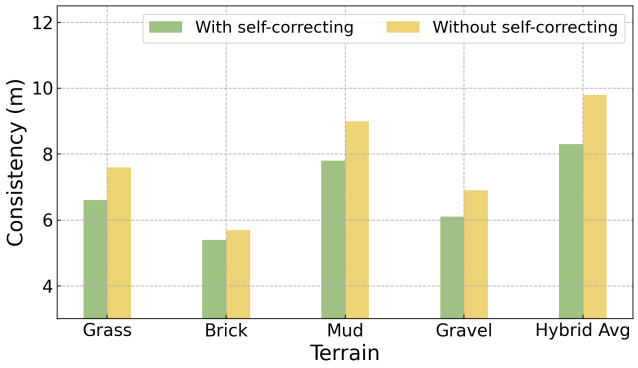

Fig. 8. The change of consistency after adding the self-correcting in different terrain, where the value of hybrid terrain have been averaged due to the diverse types. The self-correcting module improve the consistency by up to 18%.

- **Success Rate**: The success rate refers to the probability of the mobile robot reaching the target point. A trial is considered successful if the robot does not experiences collision, gets stuck, or has a roll angle greater than 30 degrees during the movement.
- **Consistency**: The L2 distance between the actual trajectory and the planned trajectory. A larger value indicates that the planned trajectory is more difficult to be executed by the tracking module.

The quantitative results are presented in Table II. Our method achieved the highest success rate, indicating the effectiveness of the proposed approach for hybrid terrain. It also shows that our method reduces the occurrence of hazardous planning by 59.37%. The traditional method achieved the best performance on smoothness, which may be attributed to the uncertainty of the neural networks. However, the traditional method fails to handle hybrid terrains effectively, resulting in the lowest success rate and poor performance in other metrics. In addition, our method achieves the best smoothness among learning-based approaches, which can be attributed to the incorporation of historical data and the utilization of a smoothness loss. The normalized trajectory of our method is the shortest, which may be because we include trajectory length as part of the training loss.

Our method and BADGR achieved the highest consistency. BADGR utilizes an end-to-end approach to learn the dynamics model of the robot, which improves the consistency. However, this improvement is limited by the quantity and the quality of the dataset. Our method uses a self-correcting structure to improve consistency, which directly evaluates whether the planned trajectory is suitable for subsequent tracking module to execute and provides a loss for training the planning network. This structure is not limited by the dataset and can improve the rationality of the network's planning results. The effectiveness of the self-correcting structure has been validated through experiments, and this structure provides an 18% enhancement in consistency.

In addition to the average data in Table II, we also analyze the success rates of different methods on different terrains to measure the performance of the different algorithms further.

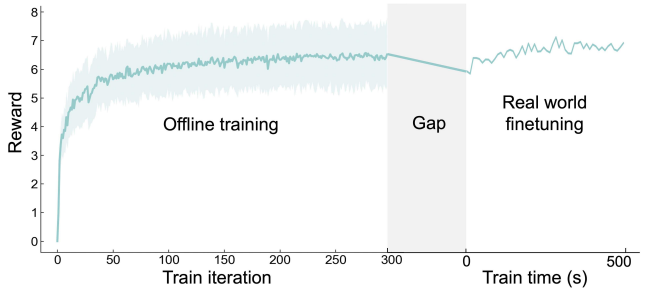

Fig. 9. The change in reward during finetuning on a test path that contains terrain that does not exist in the offline dataset.

As seen from Fig. 6, the success rate of the traditional method in the off-road environment deteriorates severely. This may be caused by the absence of terrain semantic information. The success rate of the RAORN and Terp is better than that of the traditional method, but their success rate is limited due to their imprecise extraction of terrain semantic information. The BADGR uses an end-to-end network to handle hybrid terrain, but the size of the dataset and network limits its capabilities. Our method obtains the best results. Furthermore, our method considers the impact of vibration on sensors during network training by incorporating a vibration-related reward. As shown in Fig. 7, lower vibrations of the robot are observed after executing the trajectory planned by our method.

D. Ablation Study

To analyze the impact of each module on the algorithm, we conducted ablation experiments. As shown in Table II, "No history" represents the method for removing historical observation, "No finetuning" represents the method of removing finetuning, and "No hybrid" refers to replacing the hybrid terrain semantic extraction module with a method that does not consider hybrid terrains [4], while existing methods, each terrain is assigned one type with the highest probability. The results indicate that considering hybrid terrain can effectively improve the success rate of planning.

In order to analyze the effect of finetuning, we conducted Fig. 9, which shows that the network has achieved good rewards in offline training. Meanwhile, as shown in Table II, finetuning is not necessary since our method has already surpassed the performance of the existing methods in terms of success rate without using finetuning. We also analyzed the impact of the self-correcting structure, as shown in Fig. 8, the consistency becomes smaller with self-correcting. A smaller consistency metric means the planned trajectory is more easily tracked by the subsequent tracking module, thereby enhancing the overall performance of the robot.

V. CONCLUSIONS

This work proposes a novel trajectory planning method for hybrid terrain, which is common in the natural world. We propose a terrain semantic information extraction method that can preserve hybrid terrain information. Based on this information, we design a trajectory planning network that can

generate reasonable trajectories according to the semantics and geometric information of terrain, along with the current state of the robot. To train this network, we propose a training scheme that allows offline training. This scheme includes a self-correcting structure, which makes the planned trajectory easier for subsequent tracking modules to track, thereby improving the overall capability of the robot. The future work includes studying the relationship between the distribution of different terrain types in the dataset and the training results to automatically adjust the data collection process.

REFERENCES

[1] T. H. Y. Leung, D. Ignatyev, and A. Zolotas, "Hybrid terrain traversability analysis in off-road environments," in *2022 8th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, 2022, pp. 50–56.

[2] J. Liu, X. Chen, J. Xiao, S. Lin, Z. Zheng, and H. Lu, "Hybrid map-based path planning for robot navigation in unstructured environments," *arXiv preprint arXiv:2303.05304*, 2023.

[3] X. Yin, X. Li, P. Ni, Q. Xu, and D. Kong, "A novel real-time edge-guided lidar semantic segmentation network for unstructured environments," *Remote Sensing*, vol. 15, no. 4, p. 1093, 2023.

[4] T. Guan, D. Kothandaraman, R. Chandra, A. J. Sathyamoorthy, K. Weerakoon, and D. Manocha, "Ga-nav: Efficient terrain segmentation for robot navigation in unstructured outdoor environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8138–8145, 2022.

[5] J. Hossain, A.-Z. Faridee, N. Roy, A. Basak, and D. E. Asher, "Covernav: Cover following navigation planning in unstructured outdoor environment with deep reinforcement learning," in *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*. IEEE, 2023, pp. 127–132.

[6] L. Wijayathunga, A. Rassau, and D. Chai, "Challenges and solutions for autonomous ground robot scene understanding and navigation in unstructured outdoor environments: A review," *Appl. Sci*, vol. 13, p. 9877, 2023.

[7] N. Wang, X. Li, K. Zhang, J. Wang, and D. Xie, "A survey on path planning for autonomous ground vehicles in unstructured environments," *Machines*, vol. 12, no. 1, p. 31, 2024.

[8] T. Manderson, S. Wapnick, D. Meger, and G. Dudek, "Learning to drive off road on smooth terrain in unstructured environments using an on-board camera and sparse aerial images," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1263–1269.

[9] 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.

[10] 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.

[11] A. Yang, W. Li, and Y. Hu, "Sms-mpc: Adversarial learning-based simultaneous prediction control with single model for mobile robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 905–10 912.

[12] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?" *Annual Reviews in Control*, vol. 50, pp. 233–252, 2020.

[13] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, "Graph-based subterranean exploration path planning using aerial and legged robots," *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[14] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. Velasquez, V. A. Higuti, J. Rogers, H. Tran, and G. Chowdhary, "Wayfast: Traversability predictive navigation for field robots," *CoRR*, 2022.

[15] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, no. 5, pp. 569–597, 2022.

[16] K. Zhang, F. Niroui, M. Ficocelli, and G. Nejat, "Robot navigation of environments with unknown rough terrain using deep reinforcement learning," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2018, pp. 1–7.

[17] A. J. Sathyamoorthy, K. Weerakoon, T. Guan, J. Liang, and D. Manocha, "Terrapn: Unstructured terrain navigation using online self-supervised learning," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 7197–7204.

[18] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proceedings of the Conference on Robot Learning*, vol. 100, 2020, pp. 1025–1037.

[19] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*, 2017, pp. 1126–1135.

[20] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 6750–6759.

[21] T. Z. Zhao, A. Nagabandi, K. Rakelly, C. Finn, and S. Levine, "Meld: Meta-reinforcement learning from images via latent state models," *arXiv preprint arXiv:2010.13957*, 2020.

[22] R. Dorfman, I. Shenfeld, and A. Tamar, "Offline meta learning of exploration," *arXiv preprint arXiv:2008.02598*, 2020.

[23] T. Z. Zhao, J. Luo, O. Sushkov, R. Pevceviciute, N. Heess, J. Scholz, S. Schaal, and S. Levine, "Offline meta-reinforcement learning for industrial insertion," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 6386–6393.

[24] N. T. Nguyen, L. Schilling, M. S. Angern, H. Hamann, F. Ernst, and G. Schildbach, "B-spline path planner for safe navigation of mobile robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 339–345.

[25] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," *arXiv preprint arXiv:2305.08510*, 2023.

[26] C. Sevastopoulos and S. Konstantopoulos, "A survey of traversability estimation for mobile robots," *IEEE Access*, vol. 10, pp. 96 331–96 347, 2022.

[27] S. Higa, Y. Iwashita, K. Otsu, M. Ono, O. Lamarre, A. Didier, and M. Hoffmann, "Vision-based estimation of driving energy for planetary rovers using deep learning and terramechanics," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3876–3883, 2019.

[28] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[29] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[30] S. Siva, M. Wigness, J. G. Rogers, L. Quang, and H. Zhang, "Self-reflective terrain-aware robot adaptation for consistent off-road ground navigation," *arXiv preprint arXiv:2111.06742*, 2021.

[31] 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) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154.

[32] B. Abbyasov, R. Lavrenov, A. Zakiev, K. Yakovlev, M. Svinin, and E. Magid, "Automatic tool for gazebo world construction: from a grayscale image to a 3d solid model," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7226–7232.

[33] R. Dorfman, I. Shenfeld, and A. Tamar, "Offline meta reinforcement learning–identifiability challenges and effective data collection strategies," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4607–4618, 2021.

[34] H. Zhou, P. Feng, and W. Chou, "A hybrid obstacle avoidance method for mobile robot navigation in unstructured environment," *Industrial Robot: the international journal of robotics research and application*, vol. 50, no. 1, pp. 94–106, 2023.

[35] K. Weerakoon, A. J. Sathyamoorthy, U. Patel, and D. Manocha, "Terp: Reliable planning in uneven outdoor environments using deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 9447–9453.

[36] X. Cai, M. Everett, J. Fink, and J. P. How, "Risk-aware off-road navigation via a learned speed distribution map," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2931–2937.