2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

F3DMP: Foresighted 3D Motion Planning of Mobile Robots in Wild Environments

Andong Yang, Wei Li and Yu Hu

***Abstract*— In wild environments, motion planning for mobile robots faces the challenge of local optimal path traps due to limited sensor perception range and lack of spatial awareness. Existing approaches that avoid local optimum by designing heuristic functions or high-quality global paths in wild environments are time-consuming and unstable. This work proposes F3DMP, which consists of two parts to alleviate the local optimum solution and better utilize distant terrain information. First, the entire planning framework is adapted to the three-dimensional space so that the planning result conforms to the geometric characteristics of the terrain. Second, a time allocation function based on offline reinforcement learning is proposed. This function can anticipate potential challenges or opportunities based on semantic information for the image and proactively determine a time allocation. Our planner is integrated into a complete mobile robot system and deployed to a real robot. Experiments in simulation and the real world demonstrate that our method can improve the success rate by 28% and the trajectory smoothness by 27% compared with traditional methods.**

I. INTRODUCTION

Mobile robot navigation in wild environments is essential for applications such as exploration, delivery, surveillance, etc. Motion planning is one of the kernel functions in a navigation system. In general, motion planning for ground robots aims to find a safe and low-cost path connecting a starting point and destination for mobile robots. Using optimization methods to solve motion planning is one of the most effective methods [1]. In this work, we use an optimization-based method to accomplish motion planning. The general process is as follows: first, convert a series of constraints into quadratic forms; second, use optimization-based methods to solve the optimal path according to the task objectives. This method requires providing the time allocation which is the expected travel time for each piece of the trajectory. Therefore, we propose a foresighted Offline Reinforcement Learning-based time allocation algorithm.

In traditional motion planning methods [2], the constraints include dynamic models and fixed limiting constraints such as maximum velocity and acceleration. In wild environments, terrain features can significantly affect the robot and cannot be ignored. A popular method to consider terrain information is to express terrain features through traversability, such as using neural networks to map sensor data to traversability [3], estimating traversability through the 3D map like OctoMap [4], [5]. However, these methods did not consider the z-axis information of the terrain. For example, in areas labeled as traversable, all terrain is considered the same even if there are undulations [6]. This will cause the actual distance traveled by the robot to be greater than the planned path, which further leads to inaccurate loss estimation during planning, resulting in planning failure or reduced effectiveness. There are also methods for planning directly on the 3D space [7], like point clouds [8]. However, those methods simplify the robot to a particle without considering the geometric characteristics of the robot when determining the grid traversability. To solve the above problems, we propose two improvements. First, we proposed a method to determine the traversability of a robot based on its geometric characteristics. Second, the geometry of the terrain determines three out of the six degrees of the robot pose. We use a terrain contact constraint to control the robot sticks to the ground. The motion planning in this work is built directly on the 3D OctoMap map [9]. The undulations of terrain can be reflected in the planned trajectory, thus obtaining better planning results.

The above optimization process needs to provide the time allocation of each piece of trajectory as input. In previous work, the heuristic function is used to accelerate the time

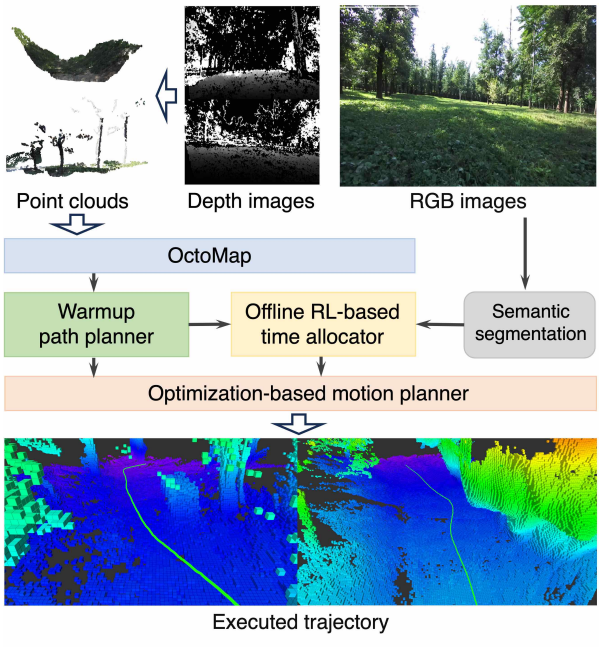

Fig. 1. This article proposes a foresighted motion planning framework for mobile robots in wild environments that directly plans in 3D space.

This work was supported by National Natural Science Foundation of China under Grant No. 62003323 and No. 62176250. 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.

allocation [10], but this method requires manually setting parameters for each scenario based on experience, which is inefficient and unstable. Adjusting the current time allocation based on the last planning results can automatically adjust the allocation results [8], [11], but this process requires multiple iterations and is not efficient. There are also method to incorporate time allocation into the optimization process [12]. However, this method only considers time allocation based on trajectory and does not consider the impact of terrain. In addition, the information obtained by the above methods is from the local three-dimensional map. Due to the heavy computational burden of mapping, the map range is usually limited.

To alleviate the above problems, we propose a foresighted time allocation method based on Offline Reinforcement Learning (Offline RL). This method can consider terrain information and proactively adjust strategies based on image data. For example, when the terrain becomes rugged, a longer time can be given to take a more conservative trajectory to improve safety. Since the information in images usually exceeds the scope of the local 3D map, this method can consider more distant terrain information than previous methods. Then, the time allocation result will be used as the input of the optimization problem so that the optimization process only needs to optimize the control points, reducing the optimization difficulty. We also use the warmup path strategy [13] to speed up the planning.

In this paper, we extend the capability of motion planning methods for mobile robots in wild environments. To summarize, our contributions are: (1) We propose a traversability judgment method that can consider the robot's geometric characteristics and a terrain contact constraint to make the entire planning framework better run in the three-dimensional space, thereby making better use of the z-axis information of the terrain. (2) we propose a foresighted time allocation method based on Offline Reinforcement Learning (Offline RL), which can consider distant terrain information. (3) The algorithm is integrated into a complete robot navigation system that includes sensing, mapping, planning, and control. We conducted experiments in the Gazebo simulator and on an actual mobile robot to evaluate the overall motion planning framework.

II. RELATED WORK

Motion Planning in the Wild Environments. One popular method is to convert terrain information into traversability, thereby transforming motion planning in wild environments into flat ground [14], [15], [16]. However, those methods develop pose in SE(2), which cannot accurately consider the changes in terrain along the z-axis direction during planning. Zhang *et al*. estimate the traversability and SE(3) pose based on lidar data. Then, the RRT* planner is used to find an initial path, and a local trajectory optimizer is proposed to find a local minimum around the initial path [17]. Krusi *et al*. employ a similar structure. A non-holonomic A* planner is used to find an initial path, and a more detailed dynamic model is used when optimizing the initial path [8]. However, to allocate proper time for each piece of the path, the above methods use a trial-and-error approach that is time-consuming. Learning-based methods are also one of the current popular directions [18]. Some methods use networks to estimate cost-map and transform the problem into traditional planning scenarios [7], [6]. However, these methods generate cost-maps only based on RGB or depth information. Zhang *et al*. propose a deep reinforcement learning network that uses raw sensory data from the robot's onboard sensors to determine a series of actions for a mobile robot to execute [19]. This learning-based method includes the complete motion planning process but requires a large amount of high-quality training data, which are hard to collect in wild environments.

Time Allocation. Traditional methods allocate the expected travel time for each trajectory in three ways: first, allocate based on a fixed velocity curve and the dynamic model [20]; second, automatically adjust the time allocation through the results of the previous planning round [8], [11]; third, add time as an optimized variable to the optimization process to improve the rationality of time allocation [21], [12]. Those methods can only consider the information in the perceptual map range, which has a limited scope, making it challenging to optimize time planning. Learning-based approaches [22], [19] establishing a mapping directly from observation to trajectory or control actions eliminates the need for a direct time allocation process. However, such methods need to interact with the environment and cannot be guaranteed safety, so the training process is challenging to carry out in the real world. Offline Reinforcement Learning is proposed to train Reinforcement Learning methods through a fixed dataset [23]. It provides a new way to solve the above problems.

III. METHODS

As shown in Fig. 2, the proposed LV3DMP runs as follows. 1) An OctoMap is built based on the depth images and pose of the mobile robot. 2) A local target is selected based on the global target and current OctoMap. Then, a warmup path from the current position to the local target is calculated by a kinodynamic planning method [24]. 3) A time allocation profile is calculated by the Offline RL-based time allocation method. 4) An optimization-based motion planner solves the optimal trajectory that satisfies the task requirements on the basis of the warmup path and time allocation. 5) The path-tracking module will execute the resulting trajectory [25]. This procedure is repeated until the mobile robot reaches the goal.

A. Motion Planning in Wild Environments

The motion planning is built based on the 3D OctoMap [26] to consider terrain changes. The mapping process is as follows. Scan the depth map to obtain point clouds, set the occupation grid based on the point cloud, and update the occupation grid every time the robot moves. Based on this map and a given global goal position. The planner selects a local target within the predefined local planning distance. To

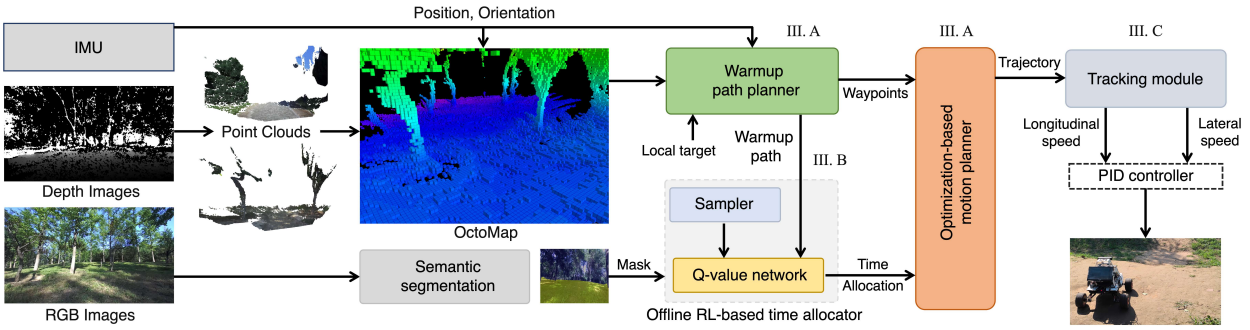

Fig. 2. The overall framework of our method. The warmup path planner plans an initial path consisting of waypoints. The semantic segmentation module outputs a classification mask of ground type based on the RGB image. The Offline RL-based time allocation module estimates a time allocation based on the mask and initial path. The optimization-based motion planner solves the final trajectory based on the prepared data. This article adapts all of the above processes to three-dimensional space to bring geometry information of the terrain into the planning process. The path-tracking module converts the trajectory into specific control commands.

accelerate the planning, we used the warmup strategy [13]. A warmup path from the current position and local target will be planned by the traditional path planning method and then replanned by the optimization process. In this work, we adopt kinodynamic planning [24] to find the warmup path. This path is a set of waypoints $\boldsymbol{w} \in \mathbb{R}^{3\times N}$, where N is the points number. This planning process requires determining the traversability of the terrain, and we used a method that takes into account the geometric characteristics of the car. We built a four-wheel model for our robot. When the height of the grid under the four wheels exceeds a specific range, the grid is judged as impassible. In addition, when sampling the following possible locations during the planning, we use a differential drive model to estimate the entire state space, not directly in Euclidean space. This allows the path to fit the robot's dynamic model.

The warmup path $\boldsymbol{w}$ is then fed to the motion planner. We use three-degree uniform B-spline [27] to represent the trajectory, and the parameters are a set of control points $\boldsymbol{q} \in \mathbb{R}^{3\times M}$. 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, $\boldsymbol{T}$ is given by the method in Sec. III-B, and the optimized parameters are the control points $\boldsymbol{q}$. The goal of this optimization is to find the $\boldsymbol{q}$* corresponding to the minimum cost:

$$\arg\min_{\boldsymbol{q}} = \int_{T_0}^{T_{M+1}} J(p(t))dt \tag{1}$$

where $J(p(t))$ is the cost function. By leveraging the convex hull property of the B-spline, we can calculate only the cost at the control points without calculating the full integration process with the same effect. From this, we define the following rewards based on control points. $C_1(\boldsymbol{q}) = \sum_{i=2}^{M-1} ||\boldsymbol{q}_{i+1} - 2\boldsymbol{q}_i + \boldsymbol{q}_{i-1}||^2$ is an elastic band cost function that simulates the elastic forces of a sequence of springs and can keep the total path length as short as possible. $C_2(\boldsymbol{q}) = \sum_{i=1}^{N} ||\boldsymbol{q}i - \boldsymbol{w}i||^2$ is the distance between warmup path and optimized path, which can constrain the optimized path to use more information from the warmup path. $C_3(\boldsymbol{q})$

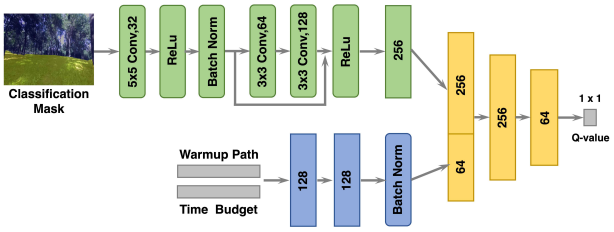

Fig. 3. The architecture of the Q-value network. Time allocation can be obtained based on the network output and warmup path.

is the safety cost, and when the height difference between two points within a certain distance on the trajectory exceeds a certain range, a penalty is given. $C_4(\boldsymbol{q})$ penalizes infeasible velocity and acceleration by giving a penalty when the velocity and acceleration exceed a certain value. $C_5(\boldsymbol{q})$ is the sum of the jerk in each control point.

In addition to the various soft constraints and rewards mentioned above, the trajectory should comply with the following hard constraints. 1) The start and end point constraints, the position, velocity, and acceleration of the start and end points need to meet pre-set values. This constraint can be expressed as $X[p(t), p'(t), p''(t)]^T = Y$, where X is a matrix composed of start time 0 and end time $\sum_{i=0}^{M} \boldsymbol{T}_i$, Y is the specific value. It is abbreviated as $\mathcal{H}(\boldsymbol{q}) = 0$ for simplicity. 2) The terrain contact constraint. Since the mobile robot can only drive on the ground, three degrees of its pose are controlled by the local geometry of the terrain. We add a terrain contact constraint, which keeps the robot position on the z-axis within a range from the ground. This constraint can be expressed as $[0, 0, 1]p(t) < H$, where H is the height of the location of the mobile robot IMU. For simplicity, it is abbreviated as $\mathcal{G}(\boldsymbol{q}) \leq 0$. So the optimization can be:

$$\begin{aligned} \arg\min_{\boldsymbol{q}} \quad & \sum_{i=1}^{5} \lambda_i C_i(\boldsymbol{q}) \\ s.t \quad & \mathcal{H}(\boldsymbol{q}) = 0 \\ & \mathcal{G}(\boldsymbol{q}) \leq 0 \end{aligned} \tag{2}$$

Due to the constraint $\mathcal{G}$ varies with changes in the ground,

the solution obtained in closed form may be unstable. In this work, we use an open-source NLopt [28] solver to solve this optimization problem.

B. Learning-Based Time Allocation

A proper time allocation is critical for optimize-based motion planning. The target of this module is to find a $\boldsymbol{T}$ for the planner to get the best $\boldsymbol{q}^*$. We model this time allocation problem as a Markov Decision Process [29]. At each moment, the agent allocates the time based on the current state. This problem can be solved using reinforcement learning. Since it is inefficient and dangerous for reinforcement learning methods to interact directly with the environment in the wild environment, we use an Offline RL method [30]. The network $f_\theta(I_{RGB}, \boldsymbol{w}, \phi)$ parameterized by θ, where I_{RGB} is the pixel classification mask in images with width W and height H, which obtained using the classification method for rugged terrain environment [31], $\boldsymbol{w}$ is the waypoints of warmup path, ϕ is the sampled action. The output is the Q-value representing the estimated reward. Select the action corresponding to the maximum reward as result. We take the long range image classification mask as one of the inputs to obtain distant terrain information which reduces local optimal planning.

The most straightforward way to define the action space in Offline RL is to define it directly as the time duration $\boldsymbol{T}$ between each control point in $\boldsymbol{q}$. However, the time slices M in $\boldsymbol{T}$ can be many. This will lead to an ample action space, which significantly increases the difficulty of training. To simplify this problem, we set the action space as the time budget $\phi = \sum_{i=0}^{M} \boldsymbol{T}_i$ for the trajectory, and then $\boldsymbol{T}$ is allocated according to the proportion of the distance between the control points to the total trajectory length.

The rewards used to train the network are defined based on the trajectory. Therefore, we first need to use the optimization-based motion planner to solve the trajectory τ based on $\boldsymbol{T}$ and the current state. For simplicity, we denote the trajectory represented by B-spline as τ_ϕ. We then define the reward as:

$$r(I_{RGB}, \boldsymbol{w}, \phi) = \sum_{i=1}^{4} \epsilon_i r_i(\tau_\phi) \tag{3}$$

where $r_1(\tau_\phi)$ is the success reward, a small positive value will be given when the robot approaches the goal and a large value when the robot reaches the goal. $r_2(\tau_\phi)$ is the vibration reward, which is the summation of the vertical motion's gradient of the robot. $r_3(\tau_\phi)$ is the length reward, which is trajectory length. $r_4(\tau_\phi)$ is the robot's average velocity.

The network structure is shown in Fig. 3. We use IQL to train the network [30]. A dataset is collected with uniform random actions and expert actions. The network is trained on this dataset with update loss:

$$L(\theta) = \mathbb{E}_{(s,\phi,s',\phi')\sim D}[L_2^\psi(r(s,\phi) + \gamma Q_{\hat{\theta}}(s',\phi') - Q_\theta(s,\phi))]$$

where $s = I_{RGB}, \boldsymbol{w}$, $s' = I'_{RGB}, \boldsymbol{w}'$ for simplicity; L_2^ψ is expectile regression [32]; D is the dataset; $Q_\theta(s,\phi)$ is the Q-value function in standard RL.

After obtaining the trained network parameters θ^*, the process of calculating the optimal time $\boldsymbol{T}^*$ is as follows: acquire the observation $s = I_{RGB}, \boldsymbol{w}$, sample the possible actions $\{\phi_1, \phi_2, ..., \phi_k\}$ from the action space, use the network to estimate the reward value $A = \{r(s,\phi_1), r(s,\phi_2), ..., r(s,\phi_k)\}$ after the execution of the sampled action, and select the action corresponding to the maximum reward value as a result $\phi^* = argmax_{\phi_i} A(\phi_i), i = 1, 2, ..., k$. Allocate $\boldsymbol{T}^*$ based on the distribution of control points and ϕ^* as described above. The general process of F3DMP is provided in Algorithm 1.

Algorithm 1 F3DMP

```
Initialize Q-value net f with trained parameters, local planning distance L, maximum iteration K, hyperparameters M, φ*, r^max.
Initialize target position S_g.
while not reach the final target and K do
    Update current state s_t, I_RGB and OctoMap O_t
    Determine next local target S'_g based on S_g, L, s_t, O_t
    Plan warmup path w
    Sample possible time budget {φ_1, ..., φ_n}
    for i=1,...,n do
        Calculate cost r = f(I_RGB, w, φ_i)
        if r > r^max then
            φ* = φ_i, r^max = r
        end if
    end for
    Obtain time allocation T* based on φ*
    Solve optimization problem to obtain control points q* based on T, O_t, S_g
    Send p(t) = B_{q*}(t), t ∈ T* to the tracking module
    Obtain control actions u_t and execute
end while
```

C. Path-tracking Module

The path-tracking controller controls the mobile robot to reach a local target. The result of our planning is a continuous curve τ_ϕ, 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 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. This allows each local target to be in a fixed range with the robot, which is convenient for the tracking module to track. If there is no intersection, the target will be set as the nearest point in the trajectory according to Euclidean distance. We define the mobile robot reaching the target as the destination within the circle.

IV. EXPERIMENTS AND RESULTS

A. Hardware Setup

1) Real-World Setup: The platform used in this work is a differential drive mobile robot (weight 26 kg; LWH 0.612m $\times$ 0.58m $\times$ 0.295m), as shown in Fig. 4 (left one). The sensors include a stereo camera ZED2i runs at 60Hz, and an

Fig. 4. The mobile robot and corresponding simulator.

Fig. 5. Examples of data collection terrain. The environment contains rocks, grass, leaves, sand, and various representative geometric features.

IMU RION TL740D runs at 100Hz. 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: We select GAZEBO [33] for simulation experiments. The terrain is built using a custom terrain generation method [19]. The mobile robot model used in GAZEBO is modeled 1:1 according to the mobile robot in the real world Fig. 4 (right one). It installed the same sensor as the real robot. We built simulation environments on a desktop computer with an Intel Xeon E5-2650 v4 processor and a Nvidia 2080Ti.

B. Dataset

The dataset used to train the network is a mixed dataset [34] collected in wild environments, as shown in Fig. 5. The dataset is collected in two ways. The first is to select a random ϕ for the planner and execute the corresponding planning result. The second is to use fixed ϕ given through our experience. This setting is to provide positive samples for the dataset to accelerate training, and these samples are not required to be optimal. The collection process is as follows. 1) Select a random goal around the robot. 2) Generate a warmup path $\boldsymbol{w}$ and terrain classification results I_{RGB}. 3) Select a ϕ and the planner output a trajectory based on $\phi, I_{RGB}, \boldsymbol{w}$. 4) Repeat 2) and 3) until the robot reaches the goal state or exceeds ten seconds and record $\{\phi, I_{RGB}, \boldsymbol{w}, r(s)\}$ as one pair of data. This process can be completed automatically, improving data collection efficiency and only requiring intervention when the robot encounters a dangerous situation. Each round of the above process can generate a trajectory. We collected 190 trajectories with random ϕ and 20 trajectories with ϕ set based on experience for a total of 19860 pairs of data.

C. Implementation

The network in Fig. 3 is trained to estimate a Q-value on a particular time budget, warmup path, and distant terrain information. The distant terrain information is a classification mask with a size of 300x375x1. Due to limited training data, we did not use a large-scale network and used residual connections to make training easier. Batch normalization is also used to increase the stability of training. To further alleviate the difficulty and data requirement for training the network, we set ϕ in $[4.0, 7.0)$ with an interval of 0.1. The lower bound represents four seconds, which is the time it takes for the mobile robot to pass through a local map at a maximum speed of 2.5m/s. The output of the Offline RL network is a discrete vector corresponding to rewards after executing each time budget ϕ. The appropriate time budget ϕ^* can be obtained by selecting the time budget ϕ corresponding to the maximum reward. The best time allocation $\boldsymbol{T}^*$, which is a vector of length 12, is allocated according to the proportion of the distance between the waypoints in the warmup path $\boldsymbol{w}$. Based on this allocation $\boldsymbol{T}^*$, we use NLopt to solve the optimization problem and obtain the optimal control points $\boldsymbol{q}^*$. The $\boldsymbol{q}^*$ corresponds to a unique B-spline τ^*, which will be sent to the subsequent tracking module for execution. We first adjust various parameters of the algorithm in the simulator and then deploy it to the actual robot. In the simulator, the classification mask I_{RGB} obtained according to the RGB image is set directly to the ground truth. The simulator environment is shown in Fig. 4.

D. Comparison to State-of-the-art

We compare our method with learning-based methods and traditional methods. DWA [35], which is a classical navigation method and uses the two-dimensional map converted from the OctoMap. Driving on Point Clouds, we abbreviate it as DPC for simplicity, uses a traditional planning structure as in this work, but take point cloud as input [8]. We conducted the same point clouds from the RGB-D camera for this method, while the other parts were not modified. Terp [22] is a learning-based approach that generates a cost-map for the environment by a Deep Reinforcement Learning (DRL) network along with an attention mask and then computes the feasible trajectory using another DRL-based method. BADGR is an end-to-end learning-based navigation system [36]. It uses reinforcement learning to map sensor data directly to control commands rather than building traversability maps and conducting traditional planning processes.

We conducted experiments on a 1.2-kilometer-long path containing rocks, grass, leaves, sand, and various representative geometric features in the real world. The following metrics quantify the performance of the algorithm.

- **Normalized Trajectory Length**: The ratio of the overall length of the actual trajectory to the straight-line distance from the start to the end point.
- **Avg Velocity**: The robot's average velocity.
- **Smoothness**: The average velocity change in all three-dimensional directions of space along the trajectory.
- **Heading Deviation**: The accumulation of the angles formed by the robot's orientation and the goal along a

TABLE I

BASELINE COMPARISONS AND ABLATION STUDY.

Method	Baseline				Our			
	DWA	**DPC**	**Terp**	**BADGR**	**No warmup path**	**Fixed time**	**No semantic**	**Our**
Avg Velocity (m/s)	0.8±0.06	1.1±0.08	0.9±0.1	1.0±0.09	0.9±0.05	1.1±0.03	1.0±0.09	**1.3 ±0.04**
Succcess Rate (%)	67±4	82±2	76±4	78±5	79±6	80±3	81±4	**86±2**
Smoothness (m/s)	1.76± 0.06	1.34±0.05	1.94±0.12	1.87± 0.10	1.52± 0.09	1.44±0.09	1.36±0.05	**1.28±0.04**
Norm. Traj. Length (-)	2.10±0.18	1.42±0.15	1.86±0.24	1.92±0.18	1.63±0.19	1.55±0.16	1.49±0.17	**1.37±0.14**
Heading Deviation (rad)	**187±15**	204±14	287±26	274±22	212±21	224± 19	208±17	194±9

Fig. 6. Trajectories planned by different algorithms in two example scenarios. Our (Green), DWA (Blue), DPC(Yellow), Terp (Grey), BADGR (Orenge)

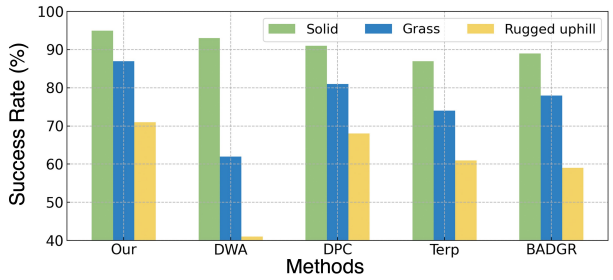

Fig. 7. Comparison of the Success Rate of different methods in different terrains.

trajectory.

- **Success Rate**: The calculation process is the same as [37]. Success is defined as reaching the destination while not getting stuck or colliding, roll angle and pitch angle of mobile robot not exceeds 30 degrees which is the upper limit of the robot's capabilities.

As shown in Table I, our method has the best Success Rate and Avg Velocity. DWA achieves the best Heading Deviation. However, DWA cannot obtain sufficient terrain information due to only considering binary maps. So, it cannot avoid dangerous terrain, resulting in a low success rate and poor smoothness. In addition, DWA detours when the terrain is judged to be impassable, resulting in a long path length. The DPC method achieved the second-highest success rate. However, because this method cannot obtain distant terrain information, it can not dynamically adjust the strategy when faced with dangerous terrain, and the planning tends to be conservative, resulting in a lower Avg Velocity, higher Heading Deviation, and longer Normalized Trajectory Length. Learning-based methods face the problem of instability, with lower Smoothness and Success Rates.

We conducted a case study to illustrate different algorithms' effectiveness further. Our method can extract distance terrain information. When the terrain changes, the time interval of control points will be adjusted by the time allocation module to reduce the possibility of local optima planning. As shown in Fig. 6, for our method, when rugged terrain appears ahead, the length of the trajectory between control points becomes longer, giving the planned trajectory more opportunities to avoid dangerous areas. At the same time, the loss of training the Offline RL network includes trajectory length, so the result will not be too conservative. The trajectory output by the DWA algorithm directly passes through dangerous terrain; Terp and BADGR is unstable and not smooth; DPC is too conservative.

E. Ablation Studies

We compare the planning performance of F3DMP with and without a warmup path, learning-based time allocation, and semantic information. As shown in Table I, the Success Rate of the method without a warmup path drops significantly, which may be caused by the difficulty in optimizing the optimal solution within the specified time. The method that employs fixed time allocation (Fixed time) gets worse results in all metrics than the one that uses the leaning-based method. The method without semantic information (No semantic) removes the semantic information extract module and sends a fixed value to the subsequent Offline RL network. This method has a worse Smoothness and lower Success Rate. This may be caused by the planner not adjusting the strategy according to the terrain ahead, resulting in planning a locally optimal trajectory, which may not reach the goal and can lead to dangerous situations.

V. CONCLUSION

This work proposes a novel motion planning method for wild environments. We propose a traversability judgment method and a terrain contact constraint to make the entire planning framework better run in the three-dimensional space. To allocate a foresighted time for planners, we propose an Offline RL-based method. This method utilizes long-range visual information to reduce the local optimal planning. The future work includes adding feedback mechanisms to score planning effects in real-time as the robot travels and recording data when lower scores are encountered. After accumulating a certain amount of data, the network can be trained again to improve performance.

REFERENCES

[1] L. Dong, Z. He, C. Song, and C. Sun, "A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures," *Journal of Systems Engineering and Electronics*, vol. 34, no. 2, pp. 439–459, 2023.

[2] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[3] 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.

[4] C. Wang, J. Wang, C. Li, D. Ho, J. Cheng, T. Yan, L. Meng, and M. Q.-H. Meng, "Safe and robust mobile robot navigation in uneven indoor environments," *Sensors*, vol. 19, no. 13, p. 2993, 2019.

[5] F. Atas, G. Cielniak, and L. Grimstad, "Elevation state-space: Surfel-based navigation in uneven environments for mobile robots," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 5715–5721.

[6] 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)*. IEEE, 2022, pp. 7197–7204.

[7] M. Sánchez-Montero, J. Morales-Rodríguez, J. L. Martinez-Rodriguez, *et al.*, "Reinforcement and curriculum learning for off-road navigation of an ugv with a 3d lidar," 2023.

[8] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017.

[9] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, pp. 189–206, 2013.

[10] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.

[11] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 2872–2879.

[12] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, *et al.*, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.

[13] B. Zhou, F. Gao, J. Pan, and S. Shen, "Robust real-time uav replanning using guided gradient-based optimization and topological paths," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1208–1214.

[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] M. Endo, T. Taniai, R. Yonetani, and G. Ishigami, "Risk-aware path planning via probabilistic fusion of traversability prediction for planetary rovers on heterogeneous terrains," *arXiv preprint arXiv:2303.01169*, 2023.

[16] 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.

[17] K. Zhang, Y. Yang, M. Fu, and M. Wang, "Traversability assessment and trajectory planning of unmanned ground vehicles with suspension systems on rough terrain," *Sensors*, vol. 19, no. 20, p. 4372, 2019.

[18] 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.

[19] 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)*. IEEE, 2018, pp. 1–7.

[20] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research: The 16th International Symposium ISRR*. Springer, 2016, pp. 649–666.

[21] J. Tordesillas, B. T. Lopez, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 1934–1940.

[22] 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.

[23] R. F. Prudencio, M. Maximo, and E. L. Colombini, "A survey on offline reinforcement learning: Taxonomy," *Review, and Open Problems*, pp. 1–21, 2022.

[24] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE international conference on robotics and automation*. IEEE, 2013, pp. 5054–5061.

[25] 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)*, 2022, pp. 10 905–10 912.

[26] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of computing*, vol. 8, no. 1, pp. 415–428, 2012.

[27] 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.

[28] S. G. Johnson and J. Schueller, "NLopt: Nonlinear optimization library," Astrophysics Source Code Library, record ascl:2111.004, p. ascl:2111.004, Nov. 2021.

[29] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[30] I. Kostrikov, A. Nair, and S. Levine, "Offline reinforcement learning with implicit q-learning," *International Conference on Learning Representations*, 2022.

[31] 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.

[32] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[33] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2004, pp. 2149–2154.

[34] K. Schweighofer, M. Hofmarcher, M.-C. Dinu, P. Renz, A. Bitto-Nemling, V. P. Patil, and S. Hochreiter, "Understanding the effects of dataset characteristics on offline reinforcement learning," in *Deep RL Workshop NeurIPS 2021*, 2021.

[35] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[36] 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.

[37] H. Karnan, K. S. Sikand, P. Atreya, S. Rabiee, X. Xiao, G. Warnell, P. Stone, and J. Biswas, "Vi-ikd: High-speed accurate off-road navigation using learned visual-inertial inverse kinodynamics," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 3294–3301.