Path Planning with Moving Obstacles

1st Xinhua Wang
Technical University of Munich
Munich, German
xinhua.wang@tum.de

2nd Yunfei Deng
Technical University of Munich
Munich, Germany
philips.deng@tum.de

***Abstract*—This paper presents a hybrid path planning method that integrates global path planning with reinforcement learning to navigate robots in complex environments with moving obstacles. We developed a grid map that includes both static and dynamic obstacles and introduced a hierarchical framework combining global guidance with a local planner. Experimental results show that our hybrid method outperforms traditional reinforcement learning approaches in dynamic environments, achieving higher success rates and better navigation performance.**

***Index Terms*—Path Planning, Reinforcement Learning, Robotics, A* Algorithm.**

I. Introduction

Path planning is a fundamental problem in robotics. It involves finding a conflict-free trajectory for a robot to move from a predefined starting point to a desired goal, given a detailed description of the environment. Traditionally, path planning is divided into two main types: offline planning and online planning. Offline planning assumes that obstacles are static and the environment is fully known, whereas online planning addresses scenarios with dynamic obstacles and partially known environments. The assumptions underlying traditional offline planning algorithms make them unsuitable for direct application to online path planning tasks due to their reliance on static obstacle conditions.

We built upon the idea of Binyu et al. [1] and propose several solutions to the problems of path planning. Our contribution are the following:

Environment Representation: We develop a grid map for a robotic system that includes both static and dynamic obstacles. The static obstacles represent fixed features within the environment, while the dynamic obstacles simulate moving objects that change position over time. This grid map serves as a fundamental component for advanced path planning and navigation algorithms, facilitating the testing and optimisation of different strategies for navigating complex and dynamic environments.

Hierarchical Framework: We introduce a hierarchical framework that combines global guidance with local reinforcement learning (RL)-based planner to support end-to-end learning in dynamic environments. The local RL planner uses spatial and temporal information within a local domain, like a field of view, to avoid potential collisions and avoid unnecessary detours. The integration of global guidance enables the robot to navigate to its goal using a fixed-size learning model, ensuring scalability even in large environments.

Reward Design: We introduce a novel reward structure that provides dense rewards without requiring the robot to strictly follow the global guidance at every step. This approach encourages the robot to explore a wider range of possible solutions.

II. Related Works

The research into classical motion planning has already been relatively mature and systematic, e.g., for graph-search-based algorithms, there are Dijkstra [2] and A* [3], for sampling-based algorithms such as Probabilistic road map (PRM) [4] and rapid-exploring random tree (RRT) [5]. However, there are several limitations to these works. From the planning architecture to the optimization problem, all of these problems constrain achieving a better planning result. With the development of Reinforcement Learning, many works combine RL with classical motion planning methods, which have addressed some bottlenecks such as long-distance navigation, dynamic obstacle interaction, etc. [6] For instance, Aleksandra Faust et al. developed an RL-RRT motion planning architecture in which a deterministic policy gradient algorithm (DDPG) agent serves as a local planner. This agent generates dynamically feasible trajectories, therefore replacing the traditional steer function within the RRT framework. [7].

III. Method

In this section, we introduce the overall structure of our model, as illustrated in Fig. 1 and present details of our approaches.

A. Environment Representation

We create a 2-dimensional discrete grid world $\mathcal{W} \subseteq \mathbb{R}^2$ with size $H \times W$ and the robot position $\mathcal{C}_r(t) = \{c_r(t)\}$. Additionally, we introduce a set of static obstacles $\mathcal{C}_s = \{s_1, \ldots, s_{N_s}\}$ with a density of $\mathcal{D}_s$ and dynamical obstacles $\mathcal{C}_d(t) = \{d_1(t), \ldots, d_{N_d}(t)\}$ with a density of $\mathcal{D}_d$.

B. Global Guidance

To enable a mobile robot to quickly reach destination c_{goal} from any starting position c_{start}, we introduce global guidance P_g. This global path is generated in a known environment containing only static obstacles. With this information, the robot receives frequent feedback signals in different scenarios, regardless of the size or complexity of the environment.

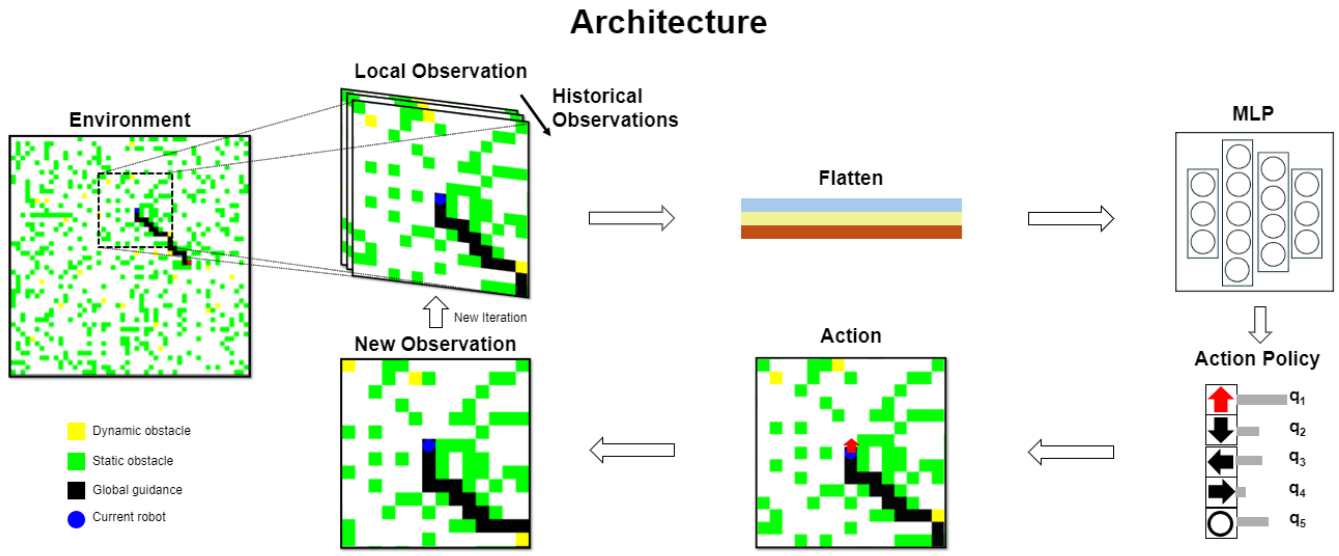


Fig. 1: The overall structure of our model.

C. Observation

Instead of using the entire map as the observation, we only consider a local field of view (FOV), which can generalize to more complex environments. To be specific, the robot gathers the local observation $O_t = \{o_t^f, o_t^s, o_t^d\}$, which encompasses the positions of free cells o_t^f, static obstacles o_t^s, and dynamic obstacles o_t^d within the local field of view (FOV) of dimensions $H_l * W_l$. Besides, we define the local segment of the global guidance P_g as the local path segment P_l, which lies within the robot's local FOV. And then, these two parts are stacked together as the observation of robots.

D. Reward Design

In reinforcement learning, the design of the reward function is crucial. To encourage the robot to quickly reach its goal along the global path while effectively avoiding dynamic obstacles, we developed an innovative reward mechanism.

At each step, the reward function offers: a large punishment $r_{collision}$ if the robot conflicts with any obstacles; a large positive reward $r_{terminal}$ if the robot reaches the target; a small penalty r_1 if the robot does not follow the global guidance; a small positive reward r_2 when the robot moves along the global guidance; a large positive reward $r_1 + N_e$ when the robot reaches one of the cells on the global guidance path, where $r_3 > |r_1| > 0$ and N_e represents the number of cells deviated from the global guidance path, measured from the point where the robot first left the path to the point where it rejoined it. The reward function can be defined formally as:

$$R(t) = \begin{cases} r_{collision} & \text{if } c_r(t+1) \in \mathcal{C}_s \cup \mathcal{C}_d(t+1) \\ r_{terminal} & \text{if } c_r(t+1) = c_{goal} \\ r_1 & \text{if } c_r(t+1) \in \mathcal{W} \setminus \mathcal{C}_s \setminus P_g \\ r_2 & \text{if } c_r(t+1) \in P_g \\ r_1 + N_e \times r_3 & \text{if } c_r(t+1) \in P_g \setminus \mathcal{C}_d(t+1) \end{cases} \quad (1)$$

E. Encoder Structure

As shown in Fig. 1, we stack three consecutive observations $\{O_t, O_{t+1}, O_{t+2}\}$, and reshape the feature into N_t one-dimensional vectors, so that the agent can exact the temporal information. Then, we use two fully connected layers to exact features, estimate the quality q_i of each state-action pair and choose the action $a_i \in \mathcal{A}$: {Up, Down, Left, Right, Idle}.

IV. Implementation

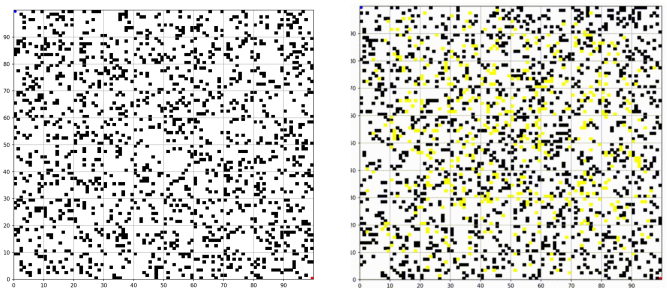
In this section, we introduce the network parameters and describe our training and testing strategies.

A. Model Parameters

In the experiments, we use the A* algorithm to generate the global guidance. The default parameters are set as follows: local FOV size $H_l = W_l = 15$, the stack of history input $N_t = 3$, the reward parameters $r_1 = -1$, $r_2 = 10$, $r_3 = 4$, $r_{collision} = -100$, and $r_{terminal} = 100$. We first flatten the input stack into 1350×1. Then we use 2 fully-connected layers, 1350×128 and 128×5 respectively, to get the next action. Between these two layers we use the ReLU activation function. For the RL algorithm, we choose the Proximal Policy Optimization (PPO) algorithm based on stable-baselines3 [8].

As shown in Fig. 2, we take two different environment maps to validate our approach, i.e., a static environment and a dynamic environment. The first one contains only static obstacles. In the dynamic environment, we set up a certain density of static obstacles and dynamic obstacles, 20% and 1% respectively. The default size of all the maps is 50×50. Dynamic obstacles are generated as robots that cannot be directly controlled and can move one cell in any direction per step. Their starting and goal positions are chosen randomly, and their paths are determined using the A* algorithm. During training and testing, each dynamic obstacle continuously

moves along its trajectory, and when motion conflict occurs, it will: 1) with a probability of 0.9, stay in its current cell until the next cell is available; 2) otherwise, reverse its direction and move back to its start cell.



(a) Static Environment (b) Dynamic Environment

Fig. 2: Static and Dynamic Environments

B. Training and Testing

We trained our model with one NVIDIA RTX 4070S GPU in Python 3.12 with PyTorch 2.3. The learning rate is fixed to 3×10^{-5}, and the training optimizer is RMSprop. We use ε-greedy to balance exploration and exploitation. The initial value is set to be $\varepsilon = 1$ and decreases to 0.1 linearly when the total training steps reach 500,000. The total training steps are 5,000,000. We trained our model three times, one in the static environment and two in dynamic environments, with global path switched on and off respectively. During testing, all methods are executed on an Intel i5-13400 CPU.

C. Performance Metrics

The following metrics are used for performance evaluation:

- **Success Rate:**

$$\text{Success Rate} = \frac{\text{Goal Reached Maps}}{\text{Total Maps}} \tag{2}$$

- **Average Reward:**
  The Average Reward is the highest average reward in one training batch.

V. Experiment

In the experiment we first evaluated the A* algorithm for global guidance planning and later effectiveness of our hybrid method and compared the success rate and average reward of the robot in different environments.

A. Astar Algorithm

As shown in Fig. 3, we successfully implemented the A* algorithm and obtained a global path from the start to the goal in the environment with static obstacles.

B. Hybrid Method

After successfully representing the environment and training with reinforcement learning, we tested the capabilities of our hybrid approach. The results are shown in the table below. For comparison, we included experiments with only static obstacles and a dynamic environment without a global path.

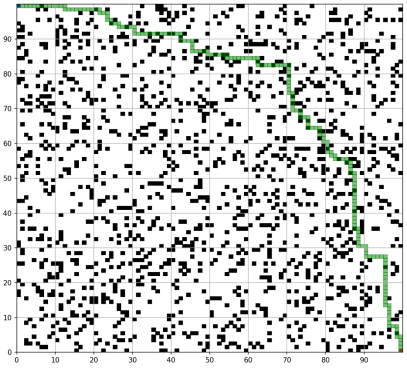


Fig. 3: Gloabal path with A* Algorithm

TABLE I: Quantitive Results

Method	Success Rate	Average Reward
Only static obstacles	79.5%	270
Dynamic environment without Global path	0%	-50.3
Hybrid Method(ours)	57.9%	244.93

1) Static Environment With A Algorithm:* In the environment with only static obstacles, our model showed its capability to follow the global guidance with a success rate of 79.5%. This indicates that the robot can learn to follow the global path with high certainty.

2) Comparison With Dynamic Environment Without Global Path: In our hybrid method, the combination of A* algorithm for global path planning and RL for real-time path planning allowed the agent to navigate efficiently through complex environments with the success rate of 57.9%, while 0% when the global path is switched off. This result shows that our hybrid method outperformed the only RL method, which highlights the strength of integrating global path planning with adaptive learning techniques to handle static and dynamic elements in the environment.

The experimental results clearly shows the advantages of our hybrid method. In dynamic environments, relying solely on RL is not sufficient for effective navigation. However, by combining global path planning using the A* algorithm with RL for local path adjustments, the robot can better handle the complexity of the environment, avoid dynamic obstacles, and achieve a higher success rate and reward. This confirms the effectiveness of integrating global path planning with adaptive learning techniques.

VI. Conclusion and Future

In this paper, we presented a hybrid path planning method that combines global path planning with local reinforcement learning to help robots navigate complex environments with moving obstacles.

Our experiments showed that the hybrid method is effective. In dynamic environments, relying only on reinforcement learning is not enough for successful navigation. However, by combining global path planning with local adjustments through reinforcement learning, the robot was better able to

handle moving obstacles, and achieve higher success rates and rewards.

Future work may focus on extending the 2D environment to 3D, and using LSTM to enhance temporal performance.

REFERENCES

[1] 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.

[2] O. Adiyatov and H. A. Varol, "A novel rrt*-based algorithm for motion planning in dynamic environments," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2017, pp. 1416–1421.

[3] Y. Lu, X. Zhang, X. Xu, and W. Yao, "Learning-based near-optimal motion planning for intelligent vehicles with uncertain dynamics," *IEEE Robotics and Automation Letters*, 2023.

[4] V. Boor, M. Overmars, and A. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, pp. 1018–1023 vol.2.

[5] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1478–1483.

[6] A. H. Qureshi, Y. Miao, A. Simeonov, and M. C. Yip, "Motion planning networks: Bridging the gap between learning-based and classical motion planners," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 48–66, 2020.

[7] H.-T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "Rl-rrt: Kinodynamic motion planning via learning reachability estimators from rl policies," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4298–4305, 2019.

[8] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html