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

Collision-Free Robot Navigation in Crowded Environments using Learning based Convex Model Predictive Control

Zhuanglei Wen, Mingze Dong, and Xiai Chen

***Abstract*— Navigating robots safely and efficiently in crowded and complex environments remains a significant challenge. However, due to the dynamic and intricate nature of these settings, planning efficient and collision-free paths for robots to track is particularly difficult. In this paper, we uniquely bridge the robot's perception, decision-making and control processes by utilizing the convex obstacle-free region computed from 2D LiDAR data. The overall pipeline is threefold: (1) We proposes a robot navigation framework that utilizes deep reinforcement learning (DRL), conceptualizing the observation as the convex obstacle-free region, a departure from general reliance on raw sensor inputs. (2) We design the action space, derived from the intersection of the robot's kinematic limits and the convex region, to enable efficient sampling of inherently collision-free reference points. These actions assists in guiding the robot to move towards the goal and interact with other obstacles during navigation. (3) We employ model predictive control (MPC) to track the trajectory formed by the reference points while satisfying constraints imposed by the convex obstacle-free region and the robot's kinodynamic limits. The effectiveness of proposed improvements has been validated through two sets of ablation studies and a comparative experiment against the Timed Elastic Band (TEB), demonstrating improved navigation performance in crowded and complex environments.**

I. INTRODUCTION

Robot navigation in crowded environments is still a challenge that has drawn significant attention from the global research community [1]. This interest is heightened by the integration of advanced learning methodologies [2], [3]. A reliable and effective autonomous navigation strategy is required due to the unpredictable dynamics. Recent advances in machine learning and DRL have facilitated the investigation of neural networks for navigation in these challenging settings [4], [5], [6].

Current DRL-based end to end navigation strategies typically define the expected speed, acceleration, and other control variables in different dimensions as actions, within continuous action spaces [7], [8]. Alternatively, these control variables are integrated into predefined actions (e.g., moving forward, braking) to form discrete action spaces [9], [10]. Nevertheless, neither continuous nor discrete actions directly depict the robot's expected trajectory. Akmandor et al. [11] proposed mapping control variables to a set of motion primitives, enabling the agent to select the optimal trajectory to track. Nevertheless, these methods still cannot guarantee the interpretability and safety of the robot's motion, losing explicit mechanism for the integration of optimization/decision and control, thereby leading to uninterpretable autonomy and impractical implementation. To address this challenge, researchers, including Gao et al. [12], have introduced innovative approaches. One effective approach is to separate the control mechanism from the neural network and employ control methods that are feasible in real-world settings.

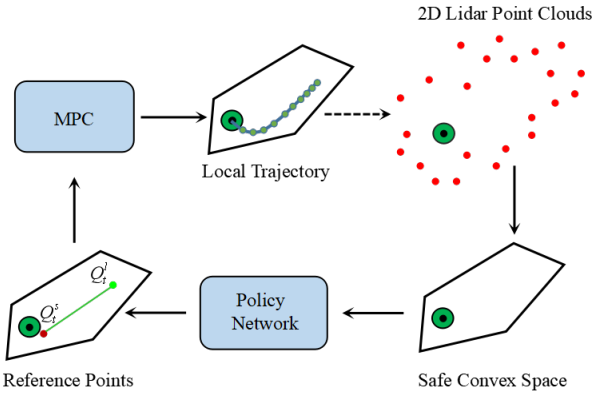

Fig. 1: Proposed Navigation Architecture: The convex obstacle-free region is obtained from 2D LiDAR point cloud data. The policy network selects reference points (Q_t^s and Q_t^l) from this convex region based on consecutive frames of observations. The MPC is then employed with online optimization to compute optimal local trajectory. The trajectory follows the reference points closely while satisfying both kinodynamic constraints and convex obstacle-free region requirements. This iterative process continues until the robot reaches its goal.

Inspired by these advancements, our study employs MPC, valued for its ability to enforce hard constraints, ensuring control commands stay within predefined limits [13]. Consequently, we define the action as a sequence of reference points for MPC's reference trajectory. Furthermore, we utilize the convex obstacle-free region derived from LiDAR point clouds as the constraint space, as depicted in Fig. 1. This strategy confines the MPC-optimized trajectory strictly within the convex region, thereby ensuring navigation that is both kinematically feasible and safe.

The principal contributions of this study are as follows:

- We design the action and action space based on the convex obstacle-free region derived from LiDAR data. To facilitate both short-term dynamic obstacle avoidance and long-term navigation, we characterize the action as a reference trajectory composed of reference points at specific time intervals. The action space is

*Corresponding author: Mingze Dong
All authors are with the College of Mechanical and Electrical Engineering, China Jiliang University, Hangzhou 310018, Zhejiang, China
1801400116@cjlu.edu.cn

defined as the intersection of the convex region with the robot's motion limits, from which we sample collision-free reference points efficiently. Subsequently, MPC is employed to track the trajectory, incorporating the convex region constraint within the MPC formulation to ensure the generation of safe and reliable control commands. Furthermore, we develop customized state space and reward function based on the convex region, reference points, and MPC-optimized trajectory.
- We implement a seven-stage curriculum training strategy and evaluate it through detailed experiments. The experiments include ablation studies to analyze the impact of varying action spaces and reward functions on navigation performance. Furthermore, a comparative evaluation with TEB method highlighted the superior performance of our method.
- We have made our work open source by publishing both the implementation and benchmark data[1].

II. RELATED WORK

Previous robot navigation approaches often consider velocity, force, potential, and other factors as key concepts [14]. However, they face challenges due to their reliance on simplistic assumptions about pedestrian dynamics. The Social Force Model (SFM) [15], along with methods like Reciprocal Velocity Obstacles (RVO) [16] and Optimal Reciprocal Collision Avoidance (ORCA) [17], provide foundational frameworks for predicting pedestrian movements, yet they may not fully capture the unpredictability of real-world environments [18]. Chen et al. [19] proposed an interactive Model Predictive Control (iMPC) framework that utilizes the iORCA model for enhanced prediction of pedestrian movements, thereby improving robot navigation in crowded environments.

Learning based methods are also applied to navigation in crowded environments. Yao et al. [20] developed an end-to-end DRL framework that uses sensor data and pedestrian maps to distinguish between obstacles and pedestrians to enhance dynamic obstacle avoidance capabilities. Hu et al. [21] presented a planning-oriented framework that utilizes Transformer-based models [22] for perception, prediction, and planning tasks. This framework overcomes the limitations of modular designs and multi-task learning, achieving end-to-end navigation.

In parallel, hybrid methods have also seen novel applications. Gao et al. [12] developed a method, encoding vehicle and predicted trajectories into binary images as the state. Action is defined as vehicle's future positions in polar coordinate. They applied convex optimization to these discrete reference points, transforming them into reachable trajectory to track. This method decouples control from other processes, with DRL focuses on suggesting reference positions and convex optimization ensuring the trajectory's feasibility for vehicle execution.

[1]https://github.com/sunnyhello369/flat_ped_sim.git

Nikdel et al. [23] introduced a hybrid method that combines DRL with classical trajectory planning, where DRL estimates human trajectories and suggests short-term goals. These goals are then used by TEB to navigate the robot in front of humans. Linh et al. [24] had explored various planning methods, including MPC, TEB, and DRL-based end-to-end navigation strategies. They designed policy network as a strategy selector, choosing the appropriate strategy based on static and dynamic environmental information. Brito et al. [25] proposed a DRL-based decision-planning method that recommends target positions for the MPC planner. Facilitate navigation by considering interactions with other agents. Compared to [25], our method does not require states of obstacles to be observed, only requires raw 2D LiDAR point cloud data.

Our navigation strategy is similar to that of Gao et al. [12], where the action is defined as a sequence of future positions for the robot. However, to achieve better navigation performance and smoother motion trajectories in dynamic crowded environments, our method utilizes the state observation based on the convex obstacle-free region. Furthermore, we integrate the convex region into the architecture of the action-state space, the reward function, and our MPC framework.

III. METHOD

A. Problem Definition

We model our problem using a Partially Observable Markov Decision Process (POMDP), which is well-suited for environments with inherent uncertainties and dynamic conditions. The POMDP model $M = (S, A, T, R, \Omega, O, \gamma)$ consists of state space S, action space A, state transition function T, reward function R, finite set of observations Ω, observation function O, and discount factor $\gamma \in [0, 1)$. At time t, the action a_t, consists of the short-term and long-term reference points, Q_t^s and Q_t^l, is formulated in Section III-C and the reward r_t detailed in Section III-F. The state s_t is defined in Section III-E.

B. Agent Dynamics

To facilitate computation in planning and control, the omni-directional mobile robot is simplified to a point mass. This simplification converts the robot's kinematics into a 2D third-order integral model. The state of the model denoted by $x = [p_x, p_y, v_x, v_y, a_x, a_y]^T$, represents the position, velocity, and acceleration. On the other hand, the control input, $u = [j_x, j_y]^T$, denotes the jerk.

The state evolution of the system can be expressed as:

$$\begin{aligned} \dot{p}_x &= v_x & \dot{p}_y &= v_y \\ \dot{v}_x &= a_x & \dot{v}_y &= a_y \\ \dot{a}_x &= j_x & \dot{a}_y &= j_y \end{aligned} \tag{1}$$

C. Action Space Design Based on Convex Region

The convex obstacle-free region is generated by the algorithm proposed by Zhong et al. [26], which can efficiently create reliable convex spaces among obstacles of any shape from LiDAR point cloud data.

To achieve efficient dynamic obstacle avoidance, the agent's action is designed as a sequence of reference points at specific time intervals. In order to simplify the training process, this sequence is simplified to two points: one that represents the robot's reference position after a control cycle (t_c) and another that represents after an MPC prediction horizon (T). These points serve as the short-term and long-term reference points, respectively. The path from the robot's current position to these points forms the reference trajectory for MPC tracking.

When designing the action space, we first ensure that the sampling of reference points is confined within the convex region. Then, centering on the robot's current position O and considering its kinematic limits, we define two circular areas. These circles, intersected with the convex region, represent our short-term and long-term action spaces, as illustrated by Fig. 2. The policy network is designed to output two sets of two-dimensional data. These data, after sigmoid activation, yield $\hat{a}_t = \left\{ \left(\alpha_t^s, \beta_t^s\right), \left(\alpha_t^l, \beta_t^l\right) | \alpha_t, \beta_t \in (0,1) \right\}$.

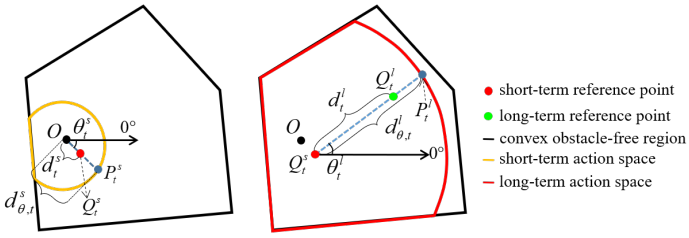

Fig. 2: Schematic of Short-Term and Long-Term Reference Point Formulation

The short-term and long-term reference points coordinates Q_t^s and Q_t^l, are calculated as Equation 2, with Fig. 2 illustrating this process. A ray at angle θ_t^s intersects with the short-term action space at P_t^s. We determine P_t^s by calculating the polar angles of the convex's vertices and finding which edge the angle θ_t^s falls within. This intersection, is solved by combining the ray and the edge equations, yields P_t^s. The distance $d_{\theta,t}^s$, from O to P_t^s, is multiplied by the scaling factor β_t^s to yield the distance d_t^s. Then we establish a new polar coordinate system centered at the short-term reference point Q_t^s. The long-term reference angle θ_t^l, is obtained by adding the increment angle $\alpha_t^l \cdot 2\pi$ to θ_t^s. Following the procedure used for the short-term point, we can determine the coordinates of Q_t^l.

$$
\begin{aligned}
\theta_t^s &= \alpha_t^s \cdot 2\pi \\
d_t^s &= \beta_t^s \cdot d_{\theta,t}^s \\
Q_t^s &= \left(d_t^s \cdot \cos\left(\theta_t^s\right), d_t^s \cdot \sin\left(\theta_t^s\right)\right) \\
\theta_t^l &= \theta_t^s + \alpha_t^l \cdot 2\pi \\
d_t^l &= \beta_t^l \cdot d_{\theta,t}^l \\
Q_t^l &= Q_t^s + \left(d_t^l \cdot \cos\left(\theta_t^l\right), d_t^l \cdot \sin\left(\theta_t^l\right)\right)
\end{aligned}
\tag{2}
$$

To navigate safely in unknown environment, our method utilizes real-time LiDAR data to construct the convex obstacle-free region for local navigation. By keeping the reference trajectory within the latest convex region, we can greatly reduce the risk of colliding with obstacles. The reference trajectory, connecting both long-term and short-term reference points is depicted by the green solid line in Fig. 3. Subsequently, the MPC framework solves the local navigation problem by calculating optimal control inputs so that the robot can continuously follow the reference trajectory. This process is iterated, ensuring the robot can follow the updated trajectory and reach the target without colliding with obstacles.

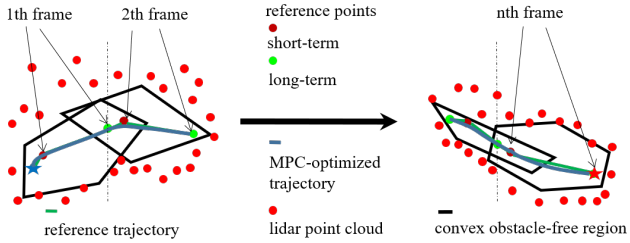

Fig. 3: Schematic of Iterative Trajectory Optimization within the Convex Region

D. Model Predictive Control Formulation

Let t_c represent the control period. If the MPC forecasts the system's state over N future control periods, the total prediction duration is $T = N \cdot t_c$. The initial state $x_{init} = \left[p_x^0, p_y^0, v_x^0, v_y^0, a_x^0, a_y^0\right]^T$ reflects the observed state at the start of the period.

After discretizing the integral model that governs robotic dynamics (as detailed in Section III-B), we derive a sequential relationship expressed as $x_i = Fx_{i-1} + Gu_{i-1}$ for $i = \{1, \ldots, N\}$. In this context, F represents the matrix for discrete-time state transitions, and G refers to the matrix for discrete-time control inputs.

Using the method mentioned in Section III-C, we get the convex region from point cloud data, with its vertices $P_j^c, j = \{1, ..., rnum_v\}$ arranged in clockwise direction. $rnum_v$ is a hyperparameter that describes the fixed number of vertices and will be introduced at Section III-E. This region constrains the MPC-optimized points to enhance safety by reducing the risk of collisions with obstacles. This constraint is efficiently met by determining whether the trajectory point Q_i is inside the convex region through the vector cross product method $\overrightarrow{Q_iP_j^c} \times \overrightarrow{Q_iP_{j+1}^c} \leq 0$.

During navigation, this vector method is also used to verify if the goal endpoint lies within the convex region. If so, this endpoint is directly used as the long-term reference point Q_t^l , which promotes early training by facilitating the acquisition of high-reward trajectories. Additionally, we introduce a final state stop cost cost_{stop}, into the MPC's cost function when the goal is within the convex region. This cost aims to minimize both velocity $V_N = \left(v_x^N, v_y^N\right)$ and acceleration $A_N = \left(a_x^N, a_y^N\right)$ at the destination, ensuring a controlled and safe termination of movement.

The short-term and long-term reference points are used to guide the positions of the first and last control period states predicted by the MPC. The cost function incorporates

discrepancies between MPC-predicted points and reference points to ensure fidelity to the reference trajectory. Ultimately, the MPC is defined as a quadratic programming problem, formulated in Equation 3, optimizing the control sequence $u^*_{0:N-1}$. w_{track} and w_{smooth} are the weights of the tracking error term and the smoothing term in the cost function respectively. w_{vend} and w_{aend} are the final state velocity and acceleration weights of the optimised cost_{stop}. The optimization process generates a sequence of MPC-optimized points $Q^*_i, i = \{1, ..., N\}$, as depicted in Fig. 4.

$$
\begin{aligned}
&\min_{u_{0:N-1}} w_{track}\left(\|Q_1 - Q_t^s\|^2 + \left\|Q_N - Q_t^l\right\|^2\right) \\
&\qquad + w_{smooth}\sum_{k=0}^{N-1}\|u_k\|^2 + \cos t_{stop} \\
&\text{cost}_{stop} = \begin{cases} w_{vend}\|V_N\|^2 \\ \quad + w_{aend}\|A_N\|^2 & \text{if goal in convex} \\ 0 & \text{otherwise} \end{cases} \\
&\text{s.t.} \quad x_0 = x_{init} \\
&\qquad x_i = Fx_{i-1} + Gu_{i-1}, \\
&\qquad \overrightarrow{Q_iP_j^c} \times \overrightarrow{Q_iP_{j+1}^c} \le 0, \\
&\qquad u_{i-1} \in \mathcal{U}, \quad \bar{x}_i \in \mathcal{S}, \\
&\qquad \forall i \in \{1, ..., N\}; \forall j = \{1, ..., rnum_v - 1\}
\end{aligned}
\tag{3}
$$

E. Observation Space Formulation

At time t, the observation for the agent o_t, as illustrated in Equation 4 and Figure 4, includes: the Euclidean distance d_t from robot to navigation goal, the velocity magnitude v_t, the angular deviation $d\theta_t$ between robot's velocity direction and the line to goal, the short-term and long-term reference points Q^s_{t-1} and Q^l_{t-1} from the previous frame's action, the MPC-optimized points Q^*_1 and Q^*_N (defined in Section III-D), and the convex region $convex_t$.

$$
o_t = \left(convex_t, d_t, v_t, d\theta_t, Q^s_{t-1}, Q^l_{t-1}, Q^*_1, Q^*_N\right) \tag{4}
$$

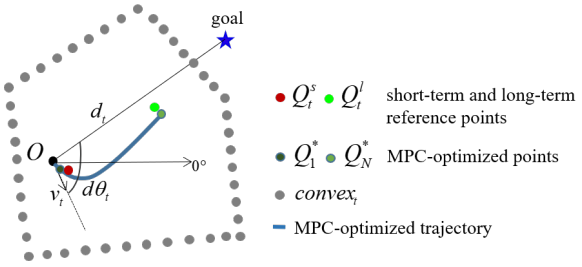

Fig. 4: Schematic of the Observation Vector

Due to the variability in shape of the convex obstacle-free region ($convex_t$) derived from different point cloud data, the number of vertices in convex polygons is not constant. This poses a challenge for neural networks, which require fixed dimension input. To address this issue, the number of vertices in the calculated convex region, denoted as num_v , is adjusted to match the predefined number $rnum_v$: vertices are added through interpolation if the count num_v is less than $rnum_v$, or reduced through sparsification if num_v exceeds $rnum_v$.

The agent can extract motion information about dynamic obstacles and its own historical trajectories from continuous frames implicitly. The agent's state, denoted as s_t, a concatenation of observations from three consecutive frames (o_{t-2}, o_{t-1}, o_t). The dimension of this state representation is set to $6rnumv + 33$. This representation contains rich spatiotemporal information that is crucial for the agent's decision making process.

F. Reward Function Formulation

The reward function plays a crucial role by setting the missions for agent. Agent's policy is formulated based on the expectation of future rewards, making reward function a critical mechanism for conveying tasks to the agent.

A positive reward $r_{success}$ is awarded to the agent when its distance to the goal d_t falls below d_{th}.

$$
r^s_t = \begin{cases} r_{success} & \text{if } d_t < d_{th}, \\ 0 & \text{otherwise.} \end{cases} \tag{5}
$$

The collision penalty r^o_t, derived from distance data $scan$ collected by LiDAR, is dynamically adjusted. The penalty decreases exponentially as the distance increases. This penalty is controlled by a positive decay factor w_{obs} and two negative terms r_{obs} and $r_{collision}$.

$$
r^o_t = \begin{cases} r_{collision} & \text{if } \min(\text{scan}) \le 0 \\ r_{obs}e^{-w_{obs}\min(\text{scan})} & \text{if } \min(\text{scan}) \le 2 \\ 0 & \text{otherwise} \end{cases} \tag{6}
$$

To mitigate sparse reward, we introduce the reward r^a_t , based on the Euclidean distance d_t between the robot and its destination which controlled by a positive factor $w_{approach}$.

$$
r^a_t = \begin{cases} 0 & \text{if } t = 1 \\ w_{approach}(d_t - d_{t-1}) & \text{otherwise} \end{cases} \tag{7}
$$

We also tried adding guide reward based on global path to r^a_t.

Short-term and long-term reference points, Q^s_t and Q^l_t, discussed in Section III-C, might not be practical to reach. However, the MPC-optimized positions Q^*_i , outlined in Section III-D, comply with kinematic constraints and provide more feasible references. Thus, to penalize the discrepancy between reference points and MPC-optimized points, we introduce r^f_t.This reward encourages the generation of reachable reference points.This penalty is controlled by two negative penalty factor $w^s_{feasible}$ and $w^l_{feasible}$.

$$
r^f_t = w^s_{feasible}\left\|Q^s_t - Q^*_1\right\|^2 + w^l_{feasible}\left\|Q^l_t - Q^*_N\right\|^2 \tag{8}
$$

Significant changes in reference points across frames can result in deviations between MPC-optimized trajectories at consecutive steps. Consequently, the agent requires increased control efforts to mitigate the influence of previous frames. To address this, the penalty r^c_t is introduced, which can also facilitate rapid learning of smooth direction changes during

the initial training phase.This penalty is controlled by two negative penalty factor w^s_{change} and w^l_{change}.

$$r^c_t = \begin{cases} 0 & \text{if } t = 1 \\ w^s_{change}\|Q^s_t - Q^s_{t-1}\|^2 & \\ + w^l_{change}\|Q^l_t - Q^l_{t-1}\|^2 & \text{otherwise} \end{cases} \tag{9}$$

Ultimately, the fixed per-step penalty, denoted as r^e_t, is introduced.

The final reward function, denoted as r_t, is composed of the above six components. These components can be adjusted or selectively ignored during phases of training.

$$r_t = r^e_t + r^s_t + r^a_t + r^o_t + r^c_t + r^f_t \tag{10}$$

G. Network Architecture

The convex obstacle-free region efficiently removes redundant information from the raw LiDAR point cloud, including noise, duplicate points, and distant obstacles. This preprocessing step allows for the implementation of a more streamlined network structure for fitting policy and value functions, as depicted in Figure 5. Given that separate networks can yield better performance in practice [27], the policy and value network are updated independently, without sharing parameters.

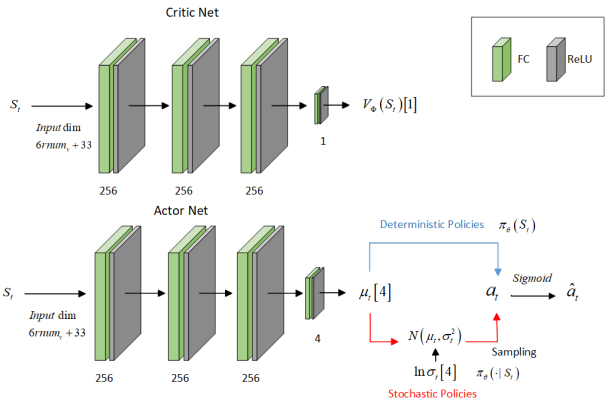

Fig. 5: Network Architecture: The value network V_ϕ and policy network π_θ are structured as four-layer fully connected networks. The policy network is augmented with a logarithmic standard deviation parameter ($\ln \sigma_t$) for each action dimension. This configuration primarily facilitates the generation of stochastic policies by allowing actions to be sampled from a Gaussian distribution.

IV. EXPERIMENTS

In this section, we introduce our training process along with the definition of seven Stages that differ in complexity (Table I). These stages are specifically designed for curriculum learning and subsequent evaluation. Following this, we evaluate our method in three parts:

(1) To demonstrate the advantages of our method in terms of action and state space design, we conduct an ablation study. This study compares our method with four different DRL navigation methods constructed with various action and state spaces (Section IV-B).

(2) To validate the effectiveness of our reward function design, we conduct another ablation study. This study conducts a comparison between our method and three other DRL navigation methods with different reward configurations (Section IV-C).

(3) In order to further analyze our method's advantages over non-DRL navigation methods in dynamic and crowded environments, we conduct a comparison between our method and the TEB [28] in test scenarios (Section IV-D).

A. Training Procedure

Our method employs a multi-stage training strategy that progresses from simple to complex scenarios, following the principle of curriculum learning. This strategy aims to enhance the agent's learning efficiency by optimizing the sequence of acquiring experience. The agent first learns basic navigation movements in an obstacle-free environment (Stage 1) initially and gradually being exposed to increasingly complex static and dynamic obstacles.

1) Stage Definition: The parameters of static and dynamic obstacles over seven Stages for staged training and subsequent evaluation are presented in Table I. Static obstacles are randomly generated polygons with three or four sides, where the maximum area of these polygons does not exceed 2 square meters. The dynamic obstacles are circular in shape.

In order to assess the performance of the trained agent, we generate 1,000 test scenarios for each Stage, from 2 to 7. These scenarios are generated by utilizing random seeds that are not included in the training dataset to configure the simulator.

TABLE I: The Parameters of Static and Dynamic Obstacles in Seven Stages for Staged Training and Evaluation

Stage	Size (m)	Static Obstacles Count	Dynamic Obstacles Count	Dynamic Obstacle Radius Range (m)	Dynamic Obstacle Speed Range (m/s)
1	20×30	0	0	-	-
2	20×30	10	0	-	-
3	20×30	10	5	0.2-0.3	0.3
4	20×30	10	10	0.2-0.3	0.3
5	10×10	0	10	0.1-0.4	0.3-0.6
6	10×10	0	20	0.1-0.4	0.3-0.6
7	10×10	0	30	0.1-0.4	0.3-0.6

2) Implementation Details: Our study utilizes the Arena-Rosnav framework [29], developed within the ROS architecture, to enable reliable robotic navigation solutions. We also adopt the implementation of Proximal Policy Optimization (PPO) [30] from Elegant-RL [31], which provides tools for network customization and training process monitoring. The simulation experiments are conducted on a high-performance PC equipped with an Intel Core i7-770to0K CPU, an Nvidia RTX 2080Ti GPU, and 32GB of RAM.In addition, we tried multi-GPU training experiments on a computer with an i7-6800k CPU and two Nvidia RTX 1080 Ti GPUs.

B. Ablation Study on State and Action Space

To verify the effectiveness of the proposed method in terms of action and state space design, we conducted comparison experiments against four methods:

Design 1: Utilizing the conventional end-to-end architecture where the observation o_t includes LiDAR data $scan_t^{'}$. Action is defined as a set of two-dimensional velocities (v_x, v_y) . The relationship between the output of the policy network (α_t, β_t) and (v_x, v_y) is detailed in (11).

$$\begin{aligned} o_t &= \left(scan_t^{'}, d_t, v_t, d\theta_t\right) \\ \hat{a}_t &= \tanh\left(a_t\right) = \{(\alpha_t, \beta_t) \,|\alpha_t, \beta_t \in (-1, 1)\} \\ v_x &= v_{min} + (v_{max} - v_{min}) \cdot \alpha_t \\ v_y &= v_{min} + (v_{max} - v_{min}) \cdot \beta_t \end{aligned} \tag{11}$$

Design 2: The state observation o_t is updated to (12), the convex obstacle-free region is taken into account. The action, consisting of (v_x, v_y), consistent with Design 1.

$$o_t = (convex_t, d_t, v_t, d\theta_t) \tag{12}$$

Design 3: The action is simplified to (Q_t^l), focusing exclusively on the long-term reference point. The calculation process of a single (Q_t^l) is similar to the short-term reference point (Q_t^s) in Section III-C, except that single (Q_t^l) is sampled from the long-term action space. In addition, Design 3 simplifies the state observation in (13) by removing terms related to Q_t^s.

$$o_t = \left(convex_t, d_t, v_t, d\theta_t, Q_{t-1}^l, Q_N^*\right) \tag{13}$$

Design 4: The observation o_t includes raw LiDAR data $scan_t^{'}$, as shown in (14). The action is still $\left(Q_t^l, Q_t^s\right)$, but calculated within the intersection of the point cloud's coverage and the robot's kinematic limits. Given that $scan_t^{'}$ is non-convex and cannot be used as a constraint for convex optimization, the constraint that the optimized trajectory must be within the convex region is removed during the MPC calculation.

$$o_t = \left(scan_t^{'}, d_t, v_t, d\theta_t, Q_{t-1}^s, Q_{t-1}^l, Q_1^*, Q_N^*\right) \tag{14}$$

After training from Stage 2 to 4, the performance of our method and Designs 1 to 4 in test scenarios is shown in Table II.

In terms of action space design, Design 3 simplifies action to a single long-term reference point. Although Design 3 has an average navigation success rate of 87.53%, close to our method's 87.93%, Design 3's average navigation time is 3.2s longer and the average navigation distance is 2.69m longer than our method in the three Stages. On the other hand, compared with Design 1 and 2, the navigation success rate of our method is 9% and 11.23% higher respectively.

In terms of state space design, compared with Design 1 and Design 4 that directly use raw LiDAR data as the observation, our method's average navigation success rate is 9% and 4.53% higher, respectively. The ablation study demonstrates the effectiveness of our method in the design of the action and state space.

TABLE II: Ablation Study on Action and State Space Designs: Comparative results of our method and Design 1 to 4 Across 1,000 test scenarios for each of the Stages from 2 to 4

Stage	Method	Success Rate (%)	Time (s)	Distance (m)	Speed (m/s)
2	Design 1	76	4.0	11.28	2.87
	Design 2	76	4.0	11.11	2.8
	Design 3	**90.3**	9.0	15.26	1.72
	Design 4	83	5.0	11.66	2.23
	Ours	89.2	5.5	12.19	2.18
3	Design 1	79.9	4.0	11.37	2.9
	Design 2	79	4.0	11.4	2.88
	Design 3	87.8	8.8	15.42	1.77
	Design 4	83.5	4.9	11.59	2.24
	Ours	**89.1**	5.9	13.04	2.17
4	Design 1	80.9	3.9	11.25	2.95
	Design 2	75.1	3.8	10.78	2.89
	Design 3	84.5	9.0	15.13	1.69
	Design 4	83.7	4.8	11.19	2.2
	Ours	**85.5**	5.8	12.51	2.1

C. Ablation Study on Reward Function

To assess the effectiveness of our reward function design, particularly the r_t^c and r_t^f items, as detailed in Section III-F, we conducted the ablation study with four different reward configurations and evaluated them across 1000 test scenarios from Stage 5 to 7. The results are presented in Table III.

The configuration of reward function r_{t1} used in our method includes all items.

$$r_{t1} = r_t^e + r_t^s + r_t^a + r_t^o + r_t^c + r_t^f \tag{15}$$

Reward function r_{t2} is based on r_{t1} but excludes the reference points change penalty (r_t^c).

$$r_{t2} = r_t^e + r_t^s + r_t^a + r_t^o + r_t^f \tag{16}$$

Reward function r_{t3} is based on r_{t1} but excludes the MPC-optimized points error penalty (r_t^f).

$$r_{t3} = r_t^e + r_t^s + r_t^a + r_t^o + r_t^c \tag{17}$$

Reward function r_{t4} excludes both the r_t^c and the r_t^f from r_{t1}.

$$r_{t4} = r_t^e + r_t^s + r_t^a + r_t^o \tag{18}$$

When evaluating the impact of composite reward functions on motion smoothness, Total Absolute Acceleration (Total Abs Acc) is a crucial metric. Total Abs Acc is the sum of the agent's absolute acceleration taken across steps within successful episodes.

The ablation study clarifies the importance of the MPC-optimized points error penalty (r_t^f) for high success rates. This is evident in the performance of r_{t1}'s performance, which achieved success rates of 87.1%, 78.1%, and 68.5% across stages. When r_t^f is excluded in r_{t3}, the success rates decline to 84.8%, 73.1%, 67.1%. Conversely, the reference points change penalty (r_t^c) slightly affects success rates, but it is significant for motion smoothness. The r_{t2} shows increased Total Abs Acc (27600.56 in Stage 5) compared to r_{t1}(26475.46 in Stage 5), indicating smoother navigation with r_t^c included.

TABLE III: Ablation Study on Reward Function Designs: Comparative results of four reward functions across 1,000 test scenarios for each of the Stage from 5 to 7

Stage	Reward	Success Rate (%)	Time (s)	Distance (m)	Speed (m/s)	Total Abs Acc
5	r_{t1}	87.1	2.9	4.59	1.57	26475.46
	r_{t2}	**87.8**	2.9	4.65	1.57	27600.56
	r_{t3}	84.8	3.4	4.38	1.40	27781.34
	r_{t4}	86.7	2.9	4.65	1.58	**26460.66**
6	r_{t1}	**78.1**	2.9	4.12	1.42	**19507.60**
	r_{t2}	77.3	2.8	4.10	1.43	20397.66
	r_{t3}	73.1	3.4	3.85	1.25	21988.81
	r_{t4}	74.6	3.0	4.35	1.46	19950.77
7	r_{t1}	68.5	3.1	4.22	1.36	6348.57
	r_{t2}	**69.9**	3.0	4.15	1.37	7218.53
	r_{t3}	67.1	3.6	3.98	1.19	6080.69
	r_{t4}	65.3	2.9	3.88	1.36	**5668.30**

D. Evaluation

Following Stages 5-7's training in tight, dynamic crowded environments, both our method and TEB method are evaluated in 1,000 test scenarios for each Stage, with results detailed in Table IV.In order to use the same kinematic model as our method, we debug and experiment on the basis of omni-directional TEB[2].

TABLE IV: Comparative Results of Our Method and Teb Method Across 1,000 Test Scenarios for Each of the Stage from 5 to 7

Stage	Method	Success Rate (%)	Time (s)	Distance (m)	Speed (m/s)
5	TEB[28]	77.1	4.4	4.64	1.11
	Ours	**87.1**	2.9	4.59	1.57
6	TEB	65.2	4.0	4.21	1.10
	Ours	**78.1**	2.9	4.12	1.42
7	TEB	57.8	3.8	3.87	1.07
	Ours	**68.5**	3.1	4.22	1.36

Our method demonstrates higher navigation success rates than the TEB method across all three Stages, outperforming TEB by 10.0%, 12.9%, and 10.7%, respectively. The results highlight the ability of our method to avoid obstacles in highly dynamic environments.

In order to compare the navigation behavior of our method with TEB, several scenarios from each stage are visualized in Figure 6. While TEB struggles with predicting the movement of dynamic obstacles, leading to potential collisions. In contrast, our method moves through available gaps, allowing for more efficient obstacle avoidance.

E. Qualitative Analysis

Figure 7 illustrates the navigation process, reflecting the iterative trajectory optimization procedure depicted in Figure 3. The robot updates the convex obstacle-free region with the latest LiDAR data. Subsequently, long-term and short-term reference points are sampled within this convex region

[2]https://github.com/pingplug/teb_local_planner/tree/omni_type

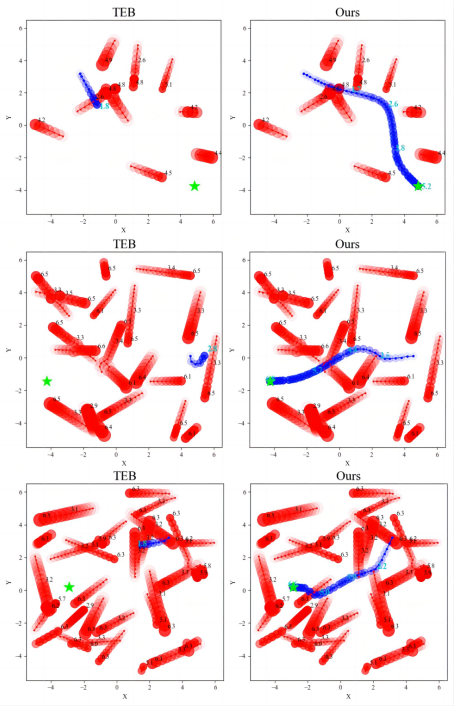

Fig. 6: Comparative Navigation Process: Our Method and TEB in Selected Test Scenarios from Stages 5, 6, 7. The three images, arranged from top to bottom, respectively depict the robot's navigation process during the test scenarios of Stages 5, 6, and 7. Dark blue line marking robot trajectory, a green star for the goal, and cyan numbers indicate the time used by robot to move to this position, in seconds. Red lines with adjacent black numbers represent dynamic obstacles' paths and timing move to this position, respectively.

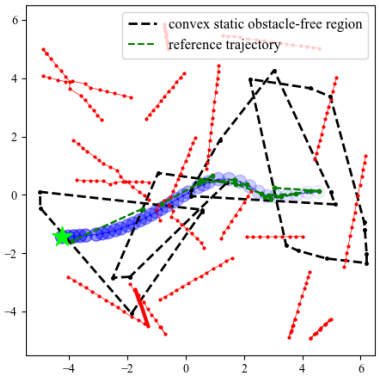

Fig. 7: Navigation Process of Our Method in a Test Scenario at Stage 6 for Qualitative Analysis: Omitting the radius and timing of dynamic obstacles, as well as the robot's timing data. The convex obstacle-free regions are indicated by black dashed lines. And a portion of reference trajectories are indicated by green dashed lines. The reference trajectory is formed by connecting long-term and short-term reference points, as detailed in Section III-C.

to form the reference trajectory. The MPC then tracks this reference trajectory to solve the local navigation problem, until the goal endpoint is within the latest convex region and is directly adopted as the long-term reference point. The process aims to promote safety by constraining both the reference trajectory and robot motion within the overlapping convex regions.

V. CONCLUSIONS

This paper introduces a DRL-based navigation framework, which consists of the design of action and state spaces, reward function and network architecture. In terms of action space design, our method integrates lidar-generated convex obstacle-free region to formulate the action space that includes both short-term and long-term reference points. For the reward function, we devise a composite reward function enriched with intermediate rewards. The experimental results conclusively demonstrate that the proposed method significantly enhances robotic performance in both static and dynamic environments, notably excelling in dynamic and crowded environments.

REFERENCES

[1] L. Kästner, X. Zhao, Z. Shen, and J. Lambrecht, "Obstacle-aware waypoint generation for long-range guidance of deep-reinforcement-learning-based navigation approaches," *arXiv preprint arXiv:2109.11639*, 2021.

[2] J. Cheng, H. Cheng, M. Q.-H. Meng, and H. Zhang, "Autonomous navigation by mobile robots in human environments: A survey," in *2018 IEEE international conference on robotics and biomimetics (ROBIO)*. IEEE, 2018, pp. 1981–1986.

[3] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[4] L. Tai and M. Liu, "A robot exploration strategy based on q-learning network," in *2016 ieee international conference on real-time computing and robotics (rcar)*. IEEE, 2016, pp. 57–62.

[5] M. Duguleana and G. Mogan, "Neural networks based reinforcement learning for mobile robots obstacle avoidance," *Expert Systems with Applications*, vol. 62, pp. 104–115, 2016.

[6] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[7] M. Pfeiffer, M. Schaeuble, J. I. Nieto, R. Y. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1527–1533, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:206852465

[8] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:15699494

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2722–2730.

[10] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3357–3364.

[11] N. Ü. Akmandor, H. Li, G. Lvov, E. Dusel, and T. Padir, "Deep reinforcement learning based robot navigation in dynamic environments using occupancy values of motion primitives," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 687–11 694.

[12] F. Gao, P. Geng, J. Guo, Y. Liu, D. Guo, Y. Su, J. Zhou, X. Wei, J. Li, and X. Liu, "Apollorl: A reinforcement learning platform for autonomous driving," *arXiv preprint arXiv:2201.12609*, 2022.

[13] B. Paden, M. Cap, 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*, p. 33–55, Mar 2016. [Online]. Available: http://dx.doi.org/10.1109/tiv.2016.2578706

[14] S. S. Samsani and M. S. Muhammad, "Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5223–5230, 2021.

[15] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[16] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE international conference on robotics and automation*. Ieee, 2008, pp. 1928–1935.

[17] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, *Reciprocal n-Body Collision Avoidance*, Jan 2011, p. 3–19. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-19457-3_1

[18] S. S. Samsani and M. S. Muhammad, "Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning," *IEEE Robotics and Automation Letters*, p. 5223–5230, Jul 2021. [Online]. Available: http://dx.doi.org/10.1109/lra.2021.3071954

[19] Y. Chen, F. Zhao, and Y. Lou, "Interactive model predictive control for robot navigation in dense crowds," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2289–2301, 2021.

[20] S. Yao, G. Chen, Q. Qiu, J. Ma, X. Chen, and J. Ji, "Crowd-aware robot navigation for pedestrians with multiple collision avoidance strategies via map-based deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8144–8150.

[21] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[23] P. Nikdel, R. Vaughan, and M. Chen, "Lbgp: Learning based goal planning for autonomous following in front," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3140–3146.

[24] K. Linh, J. Cox, T. Buiyan, J. Lambrecht *et al.*, "All-in-one: A drl-based control switch combining state-of-the-art navigation planners," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2861–2867.

[25] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: Learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4616–4623, 2021.

[26] X. Zhong, Y. Wu, D. Wang, Q. Wang, C. Xu, and F. Gao, "Generating large convex polytopes directly on point clouds," *arXiv preprint arXiv:2010.08744*, 2020.

[27] R. Raileanu and R. Fergus, "Decoupling value and policy for generalization in reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8787–8798.

[28] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *2013 European Conference on Mobile Robots*, 2013, pp. 138–143.

[29] L. Kästner, T. Buiyan, L. Jiao, T. A. Le, X. Zhao, Z. Shen, and J. Lambrecht, "Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 6456–6463.

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

[31] X.-Y. Liu, Z. Li, Z. Yang, J. Zheng, Z. Wang, A. Walid, J. Guo, and M. I. Jordan, "Elegantrl-podracer: Scalable and elastic library for cloud-native deep reinforcement learning," *NeurIPS, Workshop on Deep Reinforcement Learning*, 2021.