LLM-DiSC: LLM-enabled Distributed and Safe Coordination of Multi-Robot Systems using Control Barrier Functions

Qingqing Yang
School of Mechatronic Engineering and Automation
Shanghai University
Shanghai, China
qingqyang@shu.edu.cn

Guoxiang Zhao*
School of Future Technology
Shanghai University
Shanghai, China
gxzhao@shu.edu.cn

Xiaoqiang Ren
School of Mechatronic Engineering and Automation
Shanghai University
Shanghai, China
xqren@shu.edu.cn

Xiaofan Wang
Shanghai Institute of Technology
Shanghai, China
xfwang@shu.edu.cn

***Abstract*—Multi-robot systems (MRS) are widely used but often face motion conflicts during coordination. While large language models (LLMs) show promise in complex planning tasks, their application to MRS remains limited by hallucinations and insufficient domain knowledge. To address this, we propose LLM-DiSC, a distributed and safe coordination framework that integrates LLM-generated planners with control barrier functions (CBF). It employs centralized training with feedback from simulation and CBF-based risk evaluation to refine planning, and distributed execution for scalability. Experiments demonstrate LLM-DiSC's safety, scalability, and near-optimal performance. All supplementary materials are available at https://github.com/Yangniq/LLM-DiSC.**

***Index Terms*—large language model, multi-robot system, safe control**

I. Introduction

Multi-robot systems (MRS) have attracted significant attention thanks to their wide applications. Compared to single-robot systems, MRS can accomplish tasks more efficiently in time and is more resilient to disturbances and failures. Multi-robot motion planning is a fundamental problem in MRS and it desires to command a team of robots from initial states to their respective goal sets, while abiding dynamic constraints and environmental rules [1]. One of the challenges inside MRS is the resolution of conflicts, where conflicted robots could cause collisions or deadlocks and thereby fail to accomplish tasks. Centralized planning treats the MRS as a single entity and plans their motions in a product space, achieving completeness, safety and optimality guarantee through global optimization. However, its performance degrades as the MRS size scales up, making it suitable only for small-scale static scenarios. Such computational limitations motivate practitioners to focus on distributed planning.

Distributed planning usually enables each robot to individually compute its target navigator ignoring others and imposes a local coordinator among robots in execution. Specifically, target navigators in general fall into three types. Search-based target navigators, such as A* [2] and RRT* [3], provide near-optimal solutions given sufficient samples. Gradient-based target navigators, such as artificial potential field [4], offer efficient navigation using local information. Learning-based target navigators, such as reinforcement learning [5] or imitation learning methods [6], enable robots to learn motion policies from data. However, all these approaches face practical limitations, including high computational demands, sensitivity to local minima, reliance on expert design, or the need for large and costly training data. When it comes to local coordinators, two types are widely discussed. The first type is rank-based [7], where each robot is assigned a unique priority and lower-ranked robots yield to higher-ranked ones. This approach is efficient, but its completeness is not guaranteed. The second type relies on gradients [1]. Similar to the artificial potential field method, a distributed risk metric is introduced to assess the likelihood of collisions, prompting robots to adjust their trajectories based on the gradient of this risk value. Although scalable, this method can generate local minima, potentially leading to deadlocks. The bottleneck of distributed planning lies on the myopic reliance of collision avoidance strategy on instantaneous local information, which often leads to inefficient decisions, deadlocks, and even failures.

Control barrier functions (CBFs) have emerged as a popular tool in distributed systems to encode safety requirements [8]. CBF is a smooth function characterizing a safe set and ensures that the system always remains within this set by

* Corresponding author: Guoxiang Zhao

The work was supported in part by the National Natural Science Foundation of China under Grant 62273223, 62336005, 62403300, 62421004, and 62461160313, the Shanghai Municipal Commission of Education under Grant 24SG38, and the Aeronautical Science Fund of China under Grant 2024Z0710S6001

constraining its control input. In MRS, CBFs are extensively used for collision avoidance [9], formation maintenance [10], and consensus control [11], ensuring robots maintain safe distances while coordinating cooperative tasks and preventing mutual collisions. However, the practical implementation of CBFs still requires significant domain expertise to design and tune the CBFs. In real-world applications where such expertise may be lacking, there is a critical need to develop self-tuned approaches to reduce technical barriers for users.

Recently, pre-trained large language models (LLMs) have garnered significant attention due to their powerful contextual reasoning capabilities and flexible task transfer abilities [12]. They can translate non-formalized task descriptions into robot-executable controllers. LLMs have been employed as a controller for high-level decision making [13] and navigation control signals generation [14], as well as a controller generator for path planning [15]. Moreover, LLMs have also been used as high-level planners for path optimization [16], controller improvement [17], and deadlock resolution [18]. However, LLM's incomplete understanding of task-specific physical constraints or dynamic environments usually lead to suboptimal or impractical control policies, while the stochastic nature of their outputs could introduce safety risks in safety-critical applications. These challenges highlight the need for safeguarding validation to bridge the gap between LLM-generated strategies and real-world deployable controllers.

In this work, we develop LLM-DiSC, an LLM-enabled framework for distributed and safe coordination of multi-robot systems using control barrier functions. Our contributions are three-fold:

- LLM-CBF synergistic design: we combine LLM's contextual understanding with CBF's safety guarantees to mitigate LLM uncertainty and avoid collisions;
- Non-expert accessibility: our method enables intuitive, instruction-based control through natural language interaction and reduces the need for domain expertise;
- Reduced resource requirements: our approach overcomes common limitations by eliminating the need for expert-designed models, requiring minimal training data, and significantly reducing computational demands.

II. Problem Formulation

Consider a team of mobile robots $\mathcal{R} \triangleq \{1, 2, \ldots, N\}$ operating in a 2D plane. Each robot is modeled as a circular disk with radius $d_r > 0$. The dynamics of the robot i follow:

$$\dot{\boldsymbol{x}}_i = f(\boldsymbol{x}_i) + g(\boldsymbol{x}_i)\boldsymbol{u}_i, \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the state of robot i, $u_i \in \mathbb{R}^m$ is the control of robot i. The state x_i may vary depending on the specific dynamic model, but it invariably includes the positional information $p_i \triangleq [p_{ix},\ p_{iy}]^\top$. The environment contains multiple polygonal obstacles denoted as $\mathcal{O} \triangleq \{1, 2, \ldots, M\}$. Each robot i aims to converge to its circular target region $G_i \triangleq \{p \in \mathbb{R}^2 \mid \|p - c_i\| \le d_g\}$, defined by a center c_i and radius $d_g > 0$, while maintaining a safe distance from obstacles and other robots. Obstacles avoidance and inter-robot collision avoidance can be modeled as the following safety constraints:

$$\begin{aligned}
\mathcal{C}_{i,a}^t &= \left\{\boldsymbol{x}_i^t \in \mathbb{R}^n \;\middle|\; \|p_i^t - p_j^t\| \ge d_{safe}^a,\ \forall j \in \mathcal{R},\ j \neq i\right\}, \\
\mathcal{C}_{i,o}^t &= \left\{\boldsymbol{x}_i^t \in \mathbb{R}^n \;\middle|\; \min_{\boldsymbol{q} \in \mathcal{O}_k} \|\boldsymbol{p}_i - \boldsymbol{q}\| \ge d_{safe}^o,\ \forall k \in \mathcal{O}\right\},
\end{aligned} \tag{2}$$

where $\|\cdot\|$ is the Euclidean norm, $\mathcal{O}_k \subseteq R^2$, d_{safe}^a and d_{safe}^o respectively represent the inter-robot safety distance and the robot-obstacle safety distance, and the superscript t denotes the value of a state variable at time t. The multi-robot motion planning problem can be formulated as a constrained optimization, where each robot i must satisfy:

$$\begin{aligned}
&\lim_{t \to \infty} \boldsymbol{p}_i^t \in G_i \\
&\text{s.t.} \quad \boldsymbol{x}_i^t \in \mathcal{C}_{i,a}^t \cap \mathcal{C}_{i,o}^t = \mathcal{C}_i^t, \quad \forall t \ge 0.
\end{aligned} \tag{3}$$

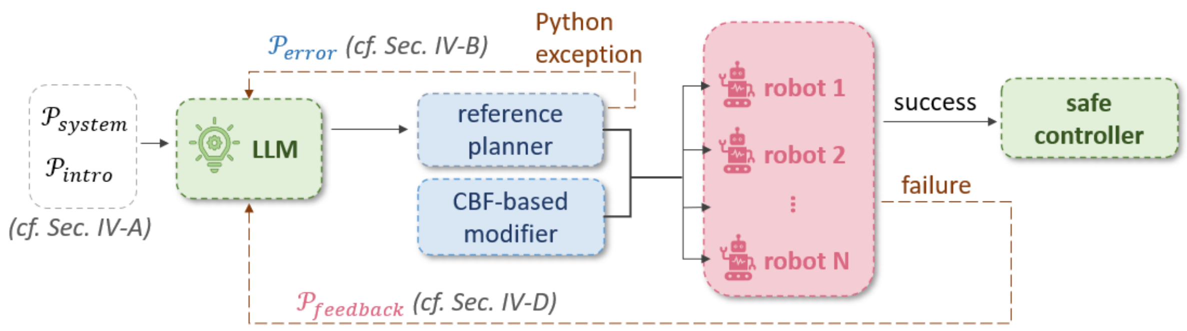
III. Method

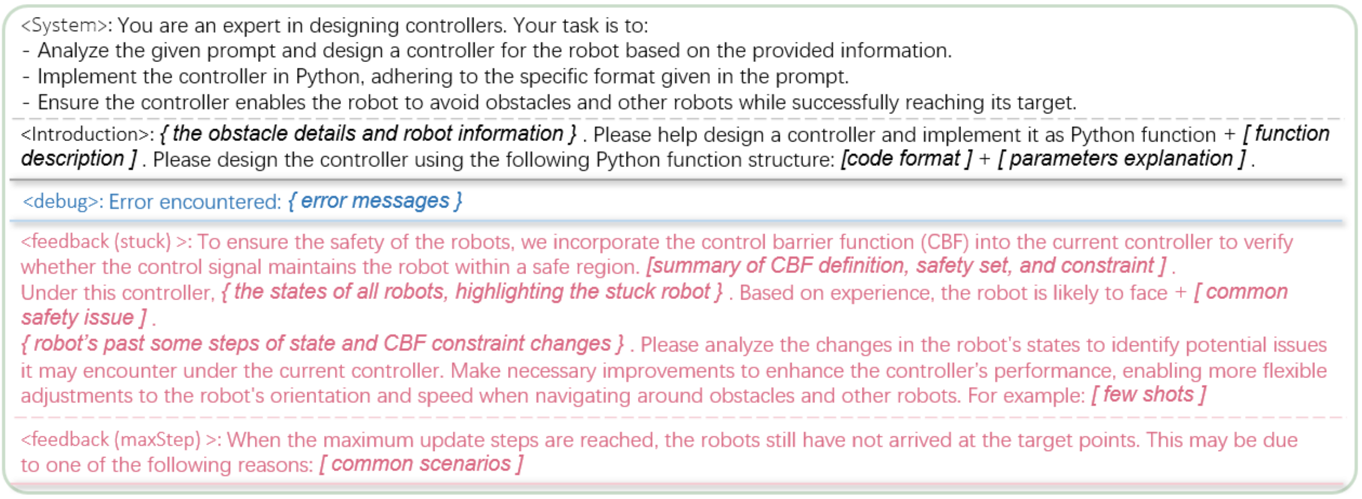
In this section, we provide a detailed explanation of our method and presents our prompt engineering. Our algorithm follows a centralized training and distributed execution framework. We design a centralized target navigator generator based on LLM, produce for each robot an individual yet uniform target navigator, and then deploy them in a distributed fashion. Simultaneously, we impose a synthesized corrector that examines the script-level semantic correctness and mission-level safety of the produced navigators to ensure the planner's reliability during execution. Additionally, we introduce a feedback optimization process to improve the planner's performance.

A. Overall Algorithm

A general overview of our method is presented in Fig. 1 and Algorithm 1. We use LLM to design a reference planner that allows the robots to avoid obstacles and reach their respective destinations. During each conversation with the LLM, the prompts consist of a system prompt $\mathcal{P}_{system}$ and several user prompts $\mathcal{P}_{user}$. The system prompt is a one-time prompt that outlines the task requirements and is usually fed to the LLM at the beginning of a conversation. A user prompt collects user's feedbacks in each interaction and could occur multiple times within a conversation. In our work, user prompt comprises of the introduction $\mathcal{P}_{intro}$, error message $\mathcal{P}_{error}$ and the feedback information $\mathcal{P}_{feedback}$, as shown in Fig. 1. Labels, manual inputs, and automatically generated content within the prompt are marked with angle brackets, square brackets, and curly brackets, respectively.

Before initiating the process, the user completes the $\mathcal{P}_{intro}$ by filling in the placeholders enclosed in square brackets with expected output descriptions. The obstacle details and robot information, including initial states, target regions, and the dynamics model, represented within curly brackets, are automatically generated (see line 1 in Alg. 1). This information, along with the $\mathcal{P}_{system}$, is fed to LLM to generate the reference planner π_{r} in Python script (see line 2).




<System>: You are an expert in designing controllers. Your task is to:
- Analyze the given prompt and design a controller for the robot based on the provided information.
- Implement the controller in Python, adhering to the specific format given in the prompt.
- Ensure the controller enables the robot to avoid obstacles and other robots while successfully reaching its target.

<Introduction>: *{ the obstacle details and robot information }* . Please help design a controller and implement it as Python function + *[function description]* . Please design the controller using the following Python function structure: *[code format]* + *[parameters explanation]* .

<debug>: Error encountered: *{ error messages }*

<feedback (stuck) >: To ensure the safety of the robots, we incorporate the control barrier function (CBF) into the current controller to verify whether the control signal maintains the robot within a safe region. *[summary of CBF definition, safety set, and constraint]* .
Under this controller, *{ the states of all robots, highlighting the stuck robot }* . Based on experience, the robot is likely to face + *[common safety issue]* .
{ robot's past some steps of state and CBF constraint changes } . Please analyze the changes in the robot's states to identify potential issues it may encounter under the current controller. Make necessary improvements to enhance the controller's performance, enabling more flexible adjustments to the robot's orientation and speed when navigating around obstacles and other robots. For example: *[few shots]*

<feedback (maxStep) >: When the maximum update steps are reached, the robots still have not arrived at the target points. This may be due to one of the following reasons: *[common scenarios]*

Fig. 1. Overview of the LLM-DiSC framework. Before starting, the user must fill in the square bracket content based on the actual situation. Once the algorithm is launched, the LLM generates a reference planner. If the combination of the reference planner and a modifier allows the MRS to safely reach the targets, the process concludes with a safe controller. If the system fails to reach the targets, relevant information is fed back to the LLM for iterative optimization of the reference planner. In case of compilation failures, error information is captured and returned to the LLM for semantic correction.

B. Semantic correction

Because of the randomness of LLM output, the generated Python script may contain semantic or logical errors, leading to compiling failures. To prevent such issues from disrupting our workflow, we designed an automated debugging process [19]. Specifically, when a Python exception occurs, we capture the error message $\mathcal{P}_{error}$, which includes details such as the exception type and traceback (see line 4) and then directly feed it back to the LLM. This guides the LLM to understand its output issues and revise the script accordingly to ensure successful execution (see lines 3-7).

C. Planner modifications

The inherent randomness of LLM outputs can also produce unsafe planners, making it necessary to introduce a safeguarding mechanism to ensure safety. Specifically, we implement a modifier π_{m} to verify and revise the generated planner. In this paper, we leverage CBF to characterize the collision risk and safeguard planner outputs. A fundamental property of CBF is the forward invariance of the controlled system, meaning that as long as the robot's initial state lies within the safe set and the control input is chosen from the safe control set, then the state remains within the safe set for all future times [8]. Unlike traditional collision avoidance methods such as Nagumo's theorem, which strictly confine trajectories at the safe set boundaries, CBFs use a Class $\mathcal{K}$ function as a buffer, allowing transient deviations while ensuring safety.

Let the safe set $\mathcal{S}$ be defined using a continuously differentiable function $h : \mathbb{R}^n \to \mathbb{R}$ as follows:

$$\mathcal{S} \triangleq \{\boldsymbol{x} \in \mathbb{R}^n : h(\boldsymbol{x}) \geq 0\}. \tag{4}$$

Definition 1: Given the system dynamics (1) of the robot i and a safe set $\mathcal{S}$, h_i is a CBF if there exists a Class $\mathcal{K}$ function α, such that $\forall \boldsymbol{x}_i \in \mathcal{S}$,

$$\sup_{\boldsymbol{u_i} \in \mathbb{U}} [L_f h_i(\boldsymbol{x}_i) + L_g h_i(\boldsymbol{x}_i)\boldsymbol{u}_i + \alpha(h_i(\boldsymbol{x}_i))] \geq 0, \tag{5}$$

where $L_f h_i$ and $L_g h_i$ denote the Lie derivatives.

Define the safe control set $\mathcal{C}_i^t$ associated with h as:

$$\mathcal{C}_i^t \triangleq \{\boldsymbol{u} \in \mathbb{R}^m : L_f h(\boldsymbol{x}_i^t) + L_g h(\boldsymbol{x}_i^t)\boldsymbol{u} + \alpha(h(\boldsymbol{x}_i^t)) \geq 0\}. \tag{6}$$

The following theorem characterizes the forward invariance of the safe set $\mathcal{S}$.

Theorem 1 ([8]): Given a CBF h with the associated set (4), any Lipschitz continuous controller $u(t) \in \mathcal{C}_i^t$ renders (4) forward invariant for (1).

The forward invariance of $\mathcal{S}$ implies that collisions are avoided so long as the system control is within the safe control set $\mathcal{C}_i^t$.

Within the context of (3), we formulate the safety indicator h according to the safety constraints (4) as:

$$\begin{aligned} h^a_{i,j}(\boldsymbol{x_i}) &= \|\boldsymbol{p}_i - \boldsymbol{p}_j\|^2 - (d^a_{safe})^2 \geq 0, \\ h^o_{i,k}(\boldsymbol{x_i}) &= \min_{\boldsymbol{q}\in\mathcal{O}_k} \|\boldsymbol{p}_i - \boldsymbol{q}\|^2 - (d^o_{safe})^2 \geq 0, \end{aligned} \tag{7}$$

where $h^a_{i,j}$ represents collision avoidance between the robots i and j, and $h^o_{i,k}$ is related to collision avoidance between the robot i and obstacle k. In order to find the safe control satisfying (6) with minimal deviation from the reference control [20], we solve the following quadratic program:

$$\begin{aligned} \min_{\boldsymbol{u}^t_i} \quad & \frac{1}{2}\|\boldsymbol{u}^t_i - \boldsymbol{u}^t_{ref}\|^2 \\ \text{s.t.} \quad & L_f h^a_{i,j}(\boldsymbol{x}^t_i) + L_g h^a_{i,j}(\boldsymbol{x}^t_i)\boldsymbol{u}^t_i + \alpha(h^a_{i,j}(\boldsymbol{x}^t_i)) \geq 0, \\ & \forall j \in \mathcal{R}\setminus\{i\}, \\ & L_f h^o_{i,k}(\boldsymbol{x}^t_i) + L_g h^o_{i,k}(\boldsymbol{x}^t_i)\boldsymbol{u}^t_i + \alpha(h^o_{i,k}(\boldsymbol{x}^t_i)) \geq 0, \\ & \forall k \in \mathcal{O}, \end{aligned} \tag{8}$$

where $\boldsymbol{u}^t_{ref} = \pi_{\mathtt{r}}(\boldsymbol{x}^t_i)$ is the output of reference planner and $\boldsymbol{u}^t_i$ is the filtered control input of $\pi_{\mathtt{m}}$. If the reference control signal $\boldsymbol{u}^t_{ref} \notin \mathcal{C}^t_i$, it is considered potentially unsafe. In such cases, the modifier computes a modified control signal by solving (8) to ensure the robot's safety (see lines 13-15).

D. Planner optimization

Constrained by (6), the reference planner may not be able to navigate each robot to its target. Therefore, it is necessary to iteratively refine the reference planner, and we achieve this by feeding back the evaluation of the synthesized planner in simulations. If the robots fail to reach their targets in simulations, we identify two possible causes: they either become `stuck` with no feasible control signal that satisfies the constraint in (6), or they exceed the maximum number of steps (`maxStep`) without all robots reaching their targets. In either case, we regard the reference planner as failed.

The robot i is considered `stuck` if it remains stationary for a predetermined number of steps, indicating an inability to navigate around obstacles or other robots while maintaining safety. In such cases, we summarize the state trajectories of all robots and the CBF values of robot i in the most recent specified number of steps. This information, denoted as $\mathcal{P}_{feedback}$, is generated by the function `FBPrompt()` and fed back to the LLM to refine and optimize the planner. Additionally, to improve success rate, we provide a few-shot example [21] that includes a concise Python implementation for obstacle avoidance along with potential optimization directions (see lines 17-23). In cases where the maximum number of steps is reached, we also supply common scenarios in $\mathcal{P}_{feedback}$ to assist the LLM in refining and optimizing the planner (see lines 25-30). This iterative process continues until all robots can reach their targets successfully.

Algorithm 1 LLM-DiSC

```
Ensure: safe controller $\pi_{\mathtt{safe}} = \pi_{\mathtt{r}} + \pi_{\mathtt{m}}$
 1: Initialize prompt:
    $\mathcal{P}_{user} \leftarrow$ UserPrompt( $\dot{x}$, $\{G_i\}_{i=1}^N$, $\{\boldsymbol{x}_i^0\}_{i=1}^N$, $\mathcal{O}$)
 2: $\pi_{\mathtt{r}} \leftarrow$ LLM.query($\mathcal{P}_{system}, \mathcal{P}_{user}$)            ▷ Section III-A
 3: while planner $\pi_{\mathtt{r}}$ fails to complie do
 4:     $\mathcal{P}_{error} \leftarrow$ captureErrorMessages()
 5:     $\mathcal{P}_{user}$.append($\mathcal{P}_{error}$)
 6:     $\pi_{\mathtt{r}} \leftarrow$ LLM.query($\mathcal{P}_{system}, \mathcal{P}_{user}$)        ▷ Section III-B
 7: end while
 8: $step \leftarrow 0$
 9: while not (exceedTokenLimit or allTargetsReached) do
10:     $step \leftarrow step + 1$
11:     for $i \in \{1, \ldots, N\}$ do
12:         $\boldsymbol{u}_i \leftarrow \pi_{\mathtt{r}}(\boldsymbol{x}_i^{step})$
13:         if $\boldsymbol{u}_i \notin \mathcal{C}_i$ then
14:             $\boldsymbol{u}_i \leftarrow \pi_{\mathtt{m}}(\boldsymbol{u}_i, \boldsymbol{x}_i^{step}, \mathcal{C}_i)$
15:         end if                                          ▷Section III-C
16:         $\boldsymbol{x}_i^{step+1} \leftarrow$ updateRobot($\boldsymbol{x}_i^{step}, \boldsymbol{u}_i$)
17:         if robot $i$ is stuck then
18:             $\mathcal{P}_{feedback} \leftarrow$ FBPrompt($\boldsymbol{x}_i^{step+1}, \{\boldsymbol{x}_j^{step}\}_{j=1}^N$)
19:             $\mathcal{P}_{user}$.append($\mathcal{P}_{feedback}$)
20:             $\pi_{\mathtt{r}} \leftarrow$ LLM.query($\mathcal{P}_{system}, \mathcal{P}_{user}$)
                                                        ▷ Section III-D
21:             $\{\boldsymbol{x}_j^0\}_{j=1}^N \leftarrow$ resetState(), $step \leftarrow 0$
22:             break
23:         end if
24:     end for
25:     if $step \geq$ maxStep then
26:         $\mathcal{P}_{feedback} \leftarrow$ FBPrompt()
27:         $\mathcal{P}_{user}$.append($\mathcal{P}_{feedback}$)
28:         $\pi_{\mathtt{r}} \leftarrow$ LLM.query($\mathcal{P}_{system}, \mathcal{P}_{user}$)   ▷ Section III-D
29:         $\{\boldsymbol{x}_j^0\}_{j=1}^N \leftarrow$ resetState(), $step \leftarrow 0$
30:     end if
31: end while
32: return $\pi_{\mathtt{safe}}$
```

IV. Experiments

We evaluated the performance of the proposed algorithm in scenarios with 5 and 10 robots[1], using both the single integrator model and the unicycle model. To assess the algorithm's feasibility, we adopted the pass@k metric [22], defined as the probability that at least one of the top k generated solutions enables all robots to safely reach their respective target regions.

[1]Additional results under different environments and numbers of robots are available at https://github.com/Yangniq/LLM-DiSC

A. Experimental Setup

All experiments share the same environment and task setup. The experimental environment consists of static polygonal obstacles and multiple robots, each with an identical radius of $d_r = 0.25$. The number of recent steps used to construct $\mathcal{P}_{feedback}$ is set to 50. We use OpenAI o1-preview as the LLM to generate the reference planner. It is worth noting that our framework is not limited to o1-preview and can be easily adapted to other LLMs by adjusting the interface.

B. Results and Analysis

*1) **Safety and Scalability**:* The experimental results validate the reliability of LLM-DiSC across different robot models and group sizes. We establish two success criteria:

- the multi-robot system remains entirely collision-free throughout its movement;
- all robots reach their respective target regions within a finite time.

We executed our framework 40 times for each case, with results presented in Table I. For the single integrator model, the **100%** success rate is achieved in the 5-robot scenario. When scaled up to 10 robots, the reliability is fully restored after an average of just **1.37** optimization iterations. These results demonstrate the strong adaptability of LLM-DiSC in coordinating multi-robot motion under simple dynamics with minimal iterations.

TABLE I
DIFFERENT ROBOT MODELS AND GROUP SIZES

Model	Size	Pass@k k=1	Pass@k k=5	Pass@k k=10	Iterations
Single Integrator	5	100%	100%	100%	1
	10	75%	100%	100%	1.37
Unicycle	5	70%	100%	100%	1.67
	10	60%	90%	95%	2.11

For the unicycle model, the nonholonomic constraints increase the complexity of motion planning, posing higher demands on LLM-DiSC. In the 5-robot scenario, the initial success rate is 70%, which rises to **100%** after an average of **1.67** optimization iterations. When scaled to 10 robots, the single-attempt success rate drops to 60%, but iterative optimization improves it to 95% with only an average of **2.11** iterations. This shows that the planner optimization mechanism effectively resolves increased conflicts as the number of robots increases, enabling the system to achieve safe coordination. Under the most challenging condition of the 10-robot unicycle model, Fig. 3 illustrates the evolution of the minimum inter-robot distance over time associated with Fig. 2, where all robots keep a safe distance from others under the guidance of LLM-DiSC. This further verifies the safety assurance capability of LLM-DiSC framework.

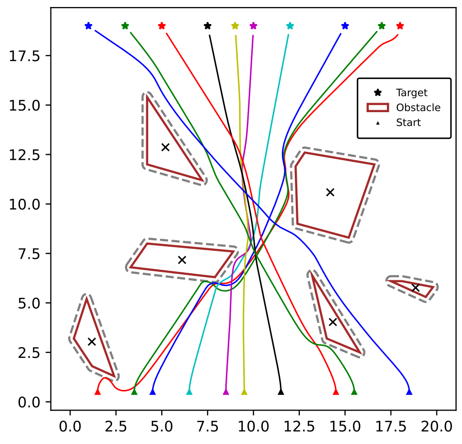


Fig. 2. Trajectories of 10 unicycle robots.

Moreover, as shown in Fig. 4, the controller generated by LLM-DiSC remains effective even under random initial positions, demonstrating its generalizability.

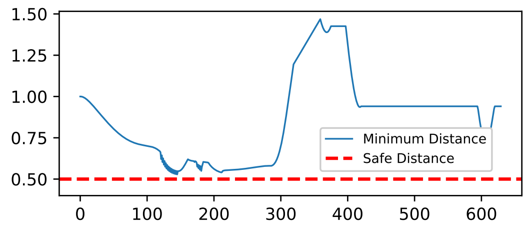


Fig. 3. Temporal evolution of minimum inter-robot distances in 10 robots.

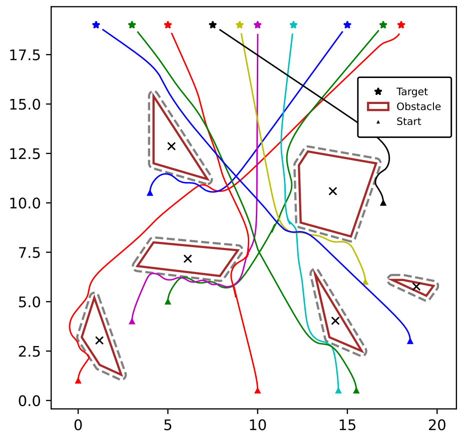


Fig. 4. Trajectories using the same controller as in Fig. 2

2) ***Near-Optimal Performance:*** In this subsection, we evaluate the near-optimality of LLM-DiSC against a conventional distributed motion planner consisting of an A* target navigator [2] combined with a Pure Pursuit tracker [23] and a local coordinator that follows the same parameterized CBF-based coordinator adopted by LLM-DiSC. When no conflict is caused among robots, A* target navigator returns the optimal solution. Both LLM-DiSC and the conventional method adopt a bounded control set $U = [-0.5, 0.5] \times [-\pi/4, \pi/4]$. Five evaluation metrics are grouped by two types: computational performance includes offline and online computation time, while execution performance are indexed by safety, traveling time, and trajectory length. Notice that the conventional distributed planner executes A* offline, whose computation time corresponds to the time required for LLM-DiSC to interact with the LLM until a safe controller is found. The online computation time for the conventional method refers to the time of running Pure Pursuit tracker with the CBF-base coordinator when the robot moves, and that for LLM-DiSC is the time to run the CBF-based coordinator when LLM-DiSC is applied. We run LLM-DiSC over 40 independent experiments to evaluate its average performance, while only run the conventional method once because of its determinism.

TABLE II
NEAR-OPTIMALITY EVALUATION

Method		LLM-DiSC	Conventional
Computations	Offline	**261.98s**	676.03s
	Online	7.85s	7.03s
Execution	Success	**Yes**	No
	Traveling Time	41.10s	40.5s
	Trajectory Length	23.25	21.69

As shown in Table II, the controller generated by LLM-DiSC successfully commands robots to safely reach their targets, whereas the conventional distributed planner fail to meet the previously defined success criteria and require repeated manual tuning of CBF parameters to do so. Although the LLM-DiSC shows slightly worse average traveling time and trajectory length compared to the conventional distributed planner, it commands all robots safely at their respective target regions with significantly reduced computation times. That is, LLM-DiSC achieves great computational efficiency at a minor sacrifice of optimality in traveling time and trajectory length; this highlights the superior responsiveness of LLM-DiSC.

*3) **Ablation Study of Different LLM Models:*** We employ o1-preview, GPT-4o and DeepSeek-r1 to assess LLM-DiSC's performance in a 10 unicycle robots scenario, with 40 runs conducted for each case.

TABLE III
ABLATION STUDY OF DIFFERENT LLMS

LLM Model		o1-preview	GPT-4o	DeepSeek-r1
Pass@k	k=1	**60%**	5%	55%
	k=5	90%	50%	**95%**
	k=10	95%	65%	**100%**
Iterations		**2.11**	4.08	2.13
Offline Computation Time		261.98s	**192s**	1069.78s
Online Computation Time		7.85s	8.86s	**6.54s**

As shown in Table III, the experimental data reveal the correlation between model characteristics and performance: GPT-4o performs relatively poorly, likely due to its general-purpose architecture's limited ability to handle the kinematic constraints of robots. In contrast, o1-preview and DeepSeek-r1, as high-intelligence reasoning models, exhibit greater advantages in addressing complex problems. In terms of computational efficiency, DeepSeek-r1's offline computation time is 1069.78 seconds, 4.1 times longer than o1-preview, while GPT-4o has the shortest computation time, but its policy quality is limited. This indicates that o1-preview offers better engineering practicality in terms of balancing quality and efficiency. This experiment validates the system is capable of adapting to different LLMs with varying characteristics. For tasks prioritizing reliability, DeepSeek-r1 can be chosen to ensure complete coverage in complex scenarios; for real-time-sensitive scenarios, o1-preview, with its balance of 261.98 seconds offline computation time and 95% success rate, becomes the optimal choice. Meanwhile, GPT-4o is currently limited by domain adaptability and requires targeted fine-tuning to enhance its practicality. This flexibility provides a theoretical basis for model selection in multi-robot systems, allowing users to optimize and balance policy quality, computational efficiency, and resource consumption based on specific needs.

V. Conclusions

In this paper, we propose a novel framework for safe, distributed multi-robot coordination. By integrating LLM reasoning with CBF safety guarantees, our approach enables collision-free navigation and efficient task execution without relying on extensive training data or expert-designed models. The experimental results validate the superior performance of LLM-DiSC across different dynamic models and group sizes. This accessible solution significantly lowers technical barriers and computational demands for multi-robot motion planning.

References

[1] G. Zhao and M. Zhu, "Scalable distributed algorithms for multi-robot near-optimal motion planning," *Automatica*, vol. 140, p. 110241, 2022.

[2] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. pearson, 2016.

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[4] C. W. Warren, "Global path planning using artificial potential fields," in *1989 IEEE International Conference on Robotics and Automation*. IEEE Computer Society, 1989, pp. 316–317.

[5] M. H. Cohen and C. Belta, "Safe exploration in model-based reinforcement learning using control barrier functions," *Automatica*, vol. 147, p. 110684, 2023.

[6] S. Choudhury, M. Bhardwaj, S. Arora, A. Kapoor, G. Ranade, S. Scherer, and D. Dey, "Data-driven planning via imitation learning," *The International Journal of Robotics Research*, vol. 37, no. 13-14, pp. 1632–1672, 2018.

[7] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 430–435.

[8] W. Xiao, C. G. Cassandras, and C. Belta, *Safe autonomy with control barrier functions: theory and applications*. Springer, 2023.

[9] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[10] J. Fu, G. Wen, X. Yu, and Z.-G. Wu, "Distributed formation navigation of constrained second-order multiagent systems with collision avoidance and connectivity maintenance," *IEEE Transactions on Cybernetics*, vol. 52, no. 4, pp. 2149–2162, 2020.

[11] C.-E. Ren, J. Zhang, and Y. Guan, "Prescribed performance bipartite consensus control for stochastic nonlinear multiagent systems under event-triggered strategy," *IEEE Transactions on Cybernetics*, vol. 53, no. 1, pp. 468–482, 2021.

[12] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, vol. 1, no. 2, 2023.

[13] W. Wang, J. Xie, C. Hu, H. Zou, J. Fan, W. Tong, Y. Wen, S. Wu, H. Deng, Z. Li, *et al.*, "Drivemlm: Aligning multi-modal large language models with behavioral planning states for autonomous driving," *arXiv preprint arXiv:2312.09245*, 2023.

[14] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K.-Y. K. Wong, Z. Li, and H. Zhao, "Drivegpt4: Interpretable end-to-end autonomous driving via large language model," *IEEE Robotics and Automation Letters*, 2024.

[15] J. Mao, J. Ye, Y. Qian, M. Pavone, and Y. Wang, "A language agent for autonomous driving," *arXiv preprint arXiv:2311.10813*, 2023.

[16] B. Yu, H. Kasaei, and M. Cao, "Co-navgpt: Multi-robot cooperative visual semantic navigation using large language models," *arXiv preprint arXiv:2310.07937*, 2023.

[17] Y. Lin, C. Li, M. Ding, M. Tomizuka, W. Zhan, and M. Althoff, "Drplanner: Diagnosis and repair of motion planners for automated vehicles using large language models," *IEEE Robotics and Automation Letters*, 2024.

[18] K. Garg *et al.*, "Large language models to the rescue: Deadlock resolution in multi-robot systems," *arXiv preprint arXiv:2404.06413*, 2024.

[19] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," *arXiv preprint arXiv:2304.05128*, 2023.

[20] X. Ding, H. Wang, Y. Ren, Y. Zheng, C. Chen, and J. He, "Online control barrier function construction for safety-critical motion control of manipulators," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2024.

[21] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[22] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[23] S. Macenski, S. Singh, F. Martín, and J. Ginés, "Regulated pure pursuit for robot path tracking," *Autonomous Robots*, vol. 47, no. 6, pp. 685–694, 2023.