Adaptive Path Planner for Swarm Robots Following a Leader

Manyung Emma Hon
Robotics Institute
Carnegie Mellon University
mehon@andrew.cmu.edu

Robert Kim
Robotics Institute
Carnegie Mellon University
yungjunk@andrew.cmu.edu

Zihan Qu
Robotics Institute
Carnegie Mellon University
zihanqu@andrew.cmu.edu

Jet Situ
Robotics Institute
Carnegie Mellon University
jets@andrew.cmu.edu

Li Wei Yang
Robotics Institute
Carnegie Mellon University
liweiy@andrew.cmu.edu

***Abstract*—This project presents an adaptive path planner for swarm robots following a leader, addressing challenges in multi-robot systems for various applications such as logistics and search-and-rescue operations. The system combines a leader-follower model with dynamic formation switching, enabling navigation and collision avoidance in complex environments. By integrating Conflict-Based Search (CBS) and Rapidly-exploring Random Tree (RRT) algorithms, the planner ensures collision-free paths for multiple robots while optimizing individual robot navigation. Key features include adaptive formation control, conflict management through CBS, and path planning using a motion primitives based RRT search. The system's effectiveness is demonstrated through experimental results in a simple 2D map environment with 8 robots and 10 obstacles, achieving a 100% collision avoidance success rate and dynamic formation switching.**

I. Introduction

Swarm robotics, inspired by collective behaviors in nature, offers a scalable and robust framework for tackling complex tasks in various domains, such as logistics, surveillance, and search-and-rescue operations. A common approach within swarm robotics is the leader-follower model, where one robot (the leader) guides a group of followers to achieve a collective goal [1], [3]. This model is particularly effective in dynamic and uncertain environments, as it simplifies coordination while allowing the swarm to adapt to changing conditions.

The leader-follower paradigm has numerous real-world applications, such as warehouse automation [5], where robots follow a leader to transport goods, or search-and-rescue missions [4], where a leading robot with advanced sensors guides others through unknown terrain. However, implementing this model in real-world scenarios presents significant challenges, particularly in maintaining swarm formations and avoiding collisions with obstacles and other robots.

This project focuses on addressing these challenges by developing an adaptive path planning system tailored to a leader-follower swarm model. The system enables effective navigation and collision avoidance while maintaining dynamic formations that can adapt to environmental constraints. A key focus of the project is to test how centralized control of the swarm can enable individual swarm robots to operate with minimal information, relying solely on their relative position to the leader and the goal position at each timestep. By leveraging a combination of Conflict-Based Search (CBS) and Rapidly-exploring Random Tree (RRT) algorithms, the planner ensures both global coordination and local adaptability. CBS facilitates collision-free paths for the swarm [2], while RRT optimizes individual robot navigation within the formation.

Key features of the system include adaptive formation control, efficient conflict resolution through CBS, and trajectory generation using a motion primitive based RRT approach. These components allow the swarm to navigate complex environments, dynamically adjust formations, and avoid collisions, ensuring effective execution of tasks in challenging scenarios.

This research contributes to the field of swarm robotics by providing a practical framework for coordinating leader-follower swarms, with potential applications in logistics, exploration, and disaster response. By enhancing the adaptability and efficiency of multi-robot systems, this project aims to bridge the gap between theoretical swarm models and real-world deployment.

II. Relevant Work

Swarm robotics has gained significant attention for its potential in applications such as surveillance, search-and-rescue, and logistics. Path planning and collision avoidance remain critical challenges, particularly for leader-follower swarm models. Existing literature provides a foundation for addressing these issues using multi-agent pathfinding (MAPF) algorithms and their extensions.

The classical MAPF problem focuses on finding conflict-free paths for multiple agents in a shared environment. Conflict-Based Search (CBS) is a prominent algorithm for solving MAPF optimally by employing a two-layered approach: a high-level search for conflicts and a low-level search for individual agent paths [2]. CBS has been extended to address practical constraints in robotics, such as kinodynamic constraints and real-time operation. For instance, the SL-CBS framework integrates state lattice with CBS to accommodate

UAV swarm dynamics and ensure spatio-temporal constraints are met [1]. Additionally, modifications like CCBS and K-CBS have expanded CBS's applicability by incorporating motion primitive conflicts and continuous-time planning [1].

Leader-follower models, a subset of swarm robotics, leverage centralized or decentralized planning techniques for swarm coordination. Centralized methods ensure global optimality but often face scalability issues, while decentralized methods prioritize computational efficiency at the cost of solution quality [1]. Hybrid approaches, such as those using CBS with enhancements like independence detection, aim to balance these trade-offs by enabling parallel conflict processing and dynamic formation adjustments [1], [2].

Moreover, a significant body of research has explored the use of advanced algorithms to address the challenges of maintaining collision-free navigation and optimal path planning in swarm robotics. For example, the integration of multi-agent reinforcement learning (MARL) into swarm systems has shown promising results in dynamically adapting to environmental changes and uncertainties. MARL-based frameworks enable decentralized decision-making, allowing individual robots to optimize their behavior locally while contributing to the swarm's global objectives [6]. Additionally, heuristic-based approaches, such as the use of evolutionary algorithms, have been investigated to improve the computational efficiency of motion planning in high-dimensional spaces, particularly for real-time applications [7].

In the context of UAV swarms, motion primitives play a critical role in addressing dynamic constraints. Algorithms like db-CBS and CB-MPC have advanced low-level planning by incorporating braking models, emergency stop primitives, and dynamic trajectory optimization [1]. However, these approaches face challenges such as computational overhead and reliance on precomputed motion primitives, highlighting the need for efficient online planners.

III. Approach

In this paper, the path planning of swarm drones in a leader-follower model is designed by utilizing a RRT planner for individual robot navigation, while CBS is employed to ensure collision-free paths for all robots. The combination of these algorithms allows the system to address both global coordination and local adaptability challenges in a dynamic swarm environment.

The system follows a hierarchical framework, with each layer contributing to a specific aspect of swarm behavior. At the top level, the framework handles Formation Control, ensuring that the swarm maintains cohesive and adaptive formations under varying environmental constraints. At the middle level, Conflict Management resolves potential collisions between robots, while the bottom level focuses on Individual Path Planning, optimizing the movement of each robot toward its assigned goal.

This layered architecture enables effective integration of global and local path planning, ensuring that the swarm can navigate complex environments, avoid obstacles, and dynamically adapt to changing conditions while maintaining the integrity of the leader-follower model. Figure 1 provides an overview of the hierarchical framework.

In addition, it is important to note that the leader in this system is not modeled as a robot but rather as a predefined trajectory. The swarm forms positions relative to the virtual leader's position at each timestep. Consequently, potential collisions between the swarm robots and the leader are not explicitly considered in this implementation. Instead, the focus remains on ensuring collision-free navigation among the swarm robots and their environment.

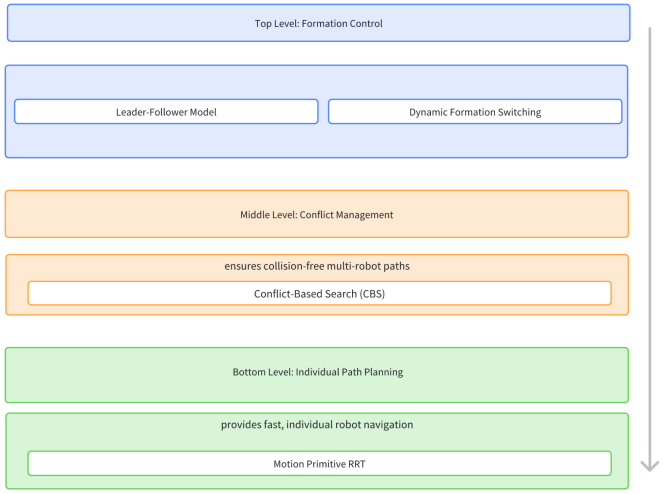

Fig. 1. High Level Architecture of Conflict Based Search Algorithm for Leader-Follower Swarm Model with Motion Primitve-based RRT

A. State, Cost, and Constraints

In the context of the hierarchical framework, the state s of the system represents the real-time configuration of all N robots in the swarm. Formally, it is defined as:

$$s = \{(x_i, y_i, \theta_i) \mid i \in \{1, 2, \ldots, N\}\}$$

Here, (x_i, y_i) denotes the 2D position of robot i, and θ_i represents its orientation. The state encapsulates the swarm's positions and orientations at a given time step, serving as the foundation for formation control, conflict resolution, and path planning.

The goal G specifies the desired configuration for the swarm, including the target positions and orientations for all robots:

$$G = \{(x_i, y_i, \theta_i) \mid i \in \{1, 2, \ldots, N\}\}$$

Each robot transitions from its current state in s to its corresponding target state in G, maintaining collision-free trajectories and adhering to the leader-follower formation constraints.

To achieve this transition, the system employs a cost function $C(s, a)$, which evaluates the feasibility and efficiency of

a state-action pair (s, a). The cost function incorporates two main components:

$$C(s,a) = \sum_{i=1}^{N} \text{dist}(s_i, G_i) + \lambda \cdot \text{collisions}(s, \text{obstacles})$$

Here:

- $\text{dist}(s_i, G_i)$: The distance of robot i from its goal configuration.
- $\text{collisions}(s, \text{obstacles})$: A penalty term for collisions with obstacles or other robots.
- λ: A weighting factor balancing path efficiency and collision avoidance.

The system also incorporates obstacle constraints to ensure safe navigation:

$$\forall i, \forall t, (x_i, y_i) \notin \text{obstacle region}$$

These constraints prevent robots from entering restricted areas and dynamically adapt to environmental changes. By integrating the state, goal definitions, cost function, and constraints, the system ensures efficient and collision-free navigation while maintaining the integrity of the leader-follower formation.

B. Individual Path Planning

For individual robot navigation within the swarm, the system employs a motion primitive-based RRT algorithm. The use of motion primitives allows the planner to generate paths that adhere to the kinematic constraints of the robots, enabling more realistic navigation.

Motion Primitives: Motion primitives are pre-defined motion patterns or actions that the robots can execute, such as moving forward, turning left, or turning right by specific angles. In this implementation, the RRT planner utilizes a set of discrete motion primitives defined by a combination of a fixed movement distance and angle changes. For example:

- Moving forward by a specified distance.
- Turning left or right by $0°$, $12°$, $24°$, $36°$, or $48°$, while simultaneously advancing forward.

These primitives constrain the robot's movements to feasible paths, considering its physical and kinematic limits.

RRT Expansion: The RRT algorithm begins at the robot's start position and iteratively expands a tree toward random target points sampled within the allowable area. Each expansion involves:

1) Selecting a random node from the configuration space.
2) Finding the nearest node in the existing tree to the random node.
3) Expanding the tree using the motion primitives to generate a set of candidate nodes.

The planner evaluates the generated nodes for feasibility by checking for collisions with obstacles and adherence to constraints. Only nodes that satisfy these conditions are added to the tree.

Goal-Oriented Sampling: To improve efficiency, the algorithm incorporates goal-oriented sampling, where a random sample has a higher probability of being the goal state. This increases the likelihood of expanding the tree in the direction of the goal, thereby accelerating convergence.

Collision Checking: Collision avoidance is a critical aspect of the RRT algorithm. Each candidate node generated during expansion is checked for collisions with obstacles, ensuring that the resulting paths are safe for execution. A buffer zone is introduced around obstacles to account for the robot's radius and motion uncertainty.

Final Path Generation: Once the tree reaches a node sufficiently close to the goal, the algorithm generates a path by backtracking through the tree from the goal node to the start node. This path is then smoothed and optimized for execution.

C. Conflict Management

The conflict management layer in this project is implemented using the CBS algorithm that ensures collision-free navigation for multiple robots by iteratively detecting and resolving conflicts. This implementation uses the following data structures to coordinate the paths of all robots.

- *Paths Dictionary:* A dictionary stores the planned trajectories for each robot, where the key is the robot's ID, and the value is a list of tuples (x, y, θ, t). Here, x and y denote the robot's position, θ its orientation, and t the time step.
- *Conflicts Dictionary:* Conflicts between robots are stored in a dictionary using robot IDs as keys. Each value is a list of conflicts, represented as tuples $(r2\_id, t)$, where $r2\_id$ is the ID of the conflicting robot, and t is the time step of the conflict.
- *Constraints Dictionary:* Constraints for each robot are stored in a dictionary. Constraints are represented as a list of $[x, y, t]$, specifying positions and times that the robot must avoid.

The conflict management process is divided into three primary components: conflict detection, constraint generation, and path replanning.

- *Conflict Detection:* This step identifies conflicts between robot trajectories. A conflict is detected when the Euclidean distance between two robots at the same time step is less than $2\times$robot radius. Conflicts are logged and sorted by robot ID and time step for efficient processing.
- *Constraint Generation:* For each detected conflict, constraints are generated for the involved robots. Constraints specify the positions and times that the robots must avoid. These constraints are stored in the constraints dictionary and passed to the path planner.
- *Path Replanning:* Robots involved in conflicts are replanned using the Individual Path Planner. The planner generates paths that respect the constraints, avoiding restricted positions while navigating toward the goal. If a valid path cannot be found, the robot remains stationary as a fallback.

- *Iterative Conflict Resolution:* The conflict detection and path replanning steps are repeated for a fixed number of iterations. If conflicts persist after all iterations, the robot with the highest number of conflicts is removed from active planning temporarily, allowing other robots to proceed.

The CBS-based conflict management layer operates within the hierarchical framework as the middle layer. It integrates individual robot paths generated by the RRT planner into a cohesive, conflict-free multi-robot plan. By dynamically resolving conflicts and enforcing constraints, this layer enables the swarm to navigate complex environments while maintaining its formation and avoiding collisions. The CBS algorithm implemented in this paper is captured in Algorithm 1.

D. Formation Control

The formation control layer is responsible for ensuring that the swarm of robots adheres to a desired formation while following the leader in a leader-follower model. This layer facilitates the dynamic adaptation of formations to accommodate environmental constraints and obstacles while maintaining the integrity of the swarm's structure. The input for this layer, including robot positions, leader trajectory, formation configurations, and obstacle locations, is provided in a text file format, allowing for easy customization and scalability.

a) Leader-Follower Integration: The leader robot generates a trajectory based on a predefined plan. At each timestep, the follower robots adjust their positions relative to the leader to maintain the specified formation. The leader's pose at a given time serves as a reference for calculating the target positions of all follower robots in the swarm.

b) Formation Specifications: The formation configuration is defined in separate files for each time interval. Each configuration specifies:

- The relative positions of the follower robots with respect to the leader.
- The orientation of each robot in the formation.
- The timestamp at which the formation becomes active.

A mapping of these configurations is stored in a dictionary where each key represents a timestamp and the corresponding value is a data structure encapsulating the formation details.

c) Dynamic Formation Updates: As the leader moves, the active formation is dynamically updated based on the current timestamp. The system checks whether a new formation is specified for the current timestep and, if so, loads the corresponding configuration file to update the follower robots' relative positions.

d) Obstacle Avoidance in Formation Control: To prevent collisions with obstacles, the target positions of follower robots are checked against the obstacle map. If a target position intersects with an obstacle, the position is adjusted to the nearest valid point outside the obstacle boundary. This adjustment is performed using a least-squares projection approach to ensure that the robot remains as close as possible to the intended target position while avoiding collisions.

Algorithm 1 Leader-Follower Swarm Model CBS Algorithm

Input : Start positions $\mathcal{S}$, Goal positions $\mathcal{G}$, Obstacles $\mathcal{O}$, Random sampling area $\mathcal{A}$, Robot radius r, Maximum attempts m

Output: Collision-free paths $\mathcal{P}$ for all robots, or failure if no solution exists

Initialize: $\mathcal{P} \leftarrow \{\}$
foreach *robot* $r_i \in \mathcal{S}$ **do**
  Plan an individual path $\mathcal{P}_{r_i}$ from start to goal using RRT
  if $\mathcal{P}_{r_i}$ *is* ***None*** **then**
    $\mathcal{P}_{r_i} \leftarrow$ stationary fallback path

Iterative Conflict Resolution:
for $i \leftarrow 1$ ***to*** m **do**
  $\mathcal{C} \leftarrow$ DetectConflicts($\mathcal{P}$, r)
  if $\mathcal{C}$ *is empty* **then**
    return $\mathcal{P}$`// Return conflict-free paths`
  $\mathcal{X} \leftarrow$ GenerateConstraints($\mathcal{C}$, $\mathcal{P}$)
  foreach *robot* r_i *with conflicts* **do**
    Replan path $\mathcal{P}_{r_i}$ using RRT with constraints $\mathcal{X}_{r_i}$
    if *Replanning fails* **then**
      $\mathcal{P}_{r_i} \leftarrow$ stationary fallback path

Handle Unresolved Conflicts:
if *conflicts remain after* m *attempts* **then**
  $\mathcal{R}_{conflicted} \leftarrow$ IdentifyRobotsWithConflicts($\mathcal{C}$)
  foreach *robot* $r_i \in \mathcal{R}_{conflicted}$ **do**
    $\mathcal{P}_{r_i} \leftarrow$ stationary fallback path
  $\mathcal{R}_{unconflicted} \leftarrow \mathcal{S} \setminus \mathcal{R}_{conflicted}$
  foreach *robot* $r_i \in \mathcal{R}_{unconflicted}$ **do**
    Replan path $\mathcal{P}_{r_i}$ considering updated priorities

e) Path Planning and Visualization: Once the target positions for all robots in the swarm are determined, the Conflict-Based Search (CBS) algorithm is invoked to generate collision-free paths for each robot. The CBS planner ensures that the swarm maintains its formation while safely navigating through the environment.

To facilitate debugging and analysis, a visualizer is used to render the swarm's movement in real-time. The visualizer supports dynamic updates of formations and provides animations of the swarm's movement, which can be saved as video files for further examination.

f) Algorithmic Workflow: The formation control process is summarized as follows:

1) Load the leader's plan and formation configuration files.
2) For each timestep:
   a) Compute the target positions for follower robots based on the leader's pose and the active formation.
   b) Adjust the target positions to avoid obstacles.
   c) Invoke the CBS planner to compute collision-free paths for the swarm.
3) Update the robots' states based on the planned paths.
4) Visualize the swarm's movement and save the results.

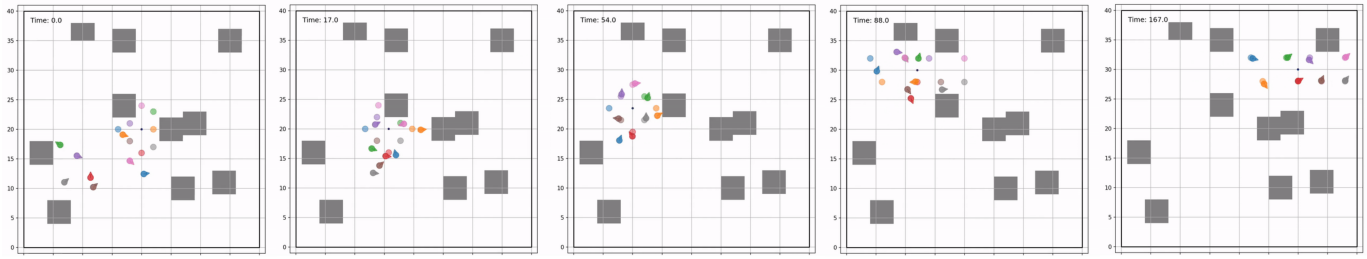

Fig. 2. Sequential snapshots of the swarm robots adapting to a dynamic formation change. Dark-colored markers represent the current positions of the robots, while light-shaded markers indicate the goal positions defined by the formation relative to the leader's position. The leader, depicted as a black dot in the center, follows a predefined trajectory, guiding the swarm through the environment. The formation change occurs at timestep $t = 80.0$, and the swarm successfully achieves the new desired configuration by $t = 167.0$, while avoiding both self-collisions and external collisions.

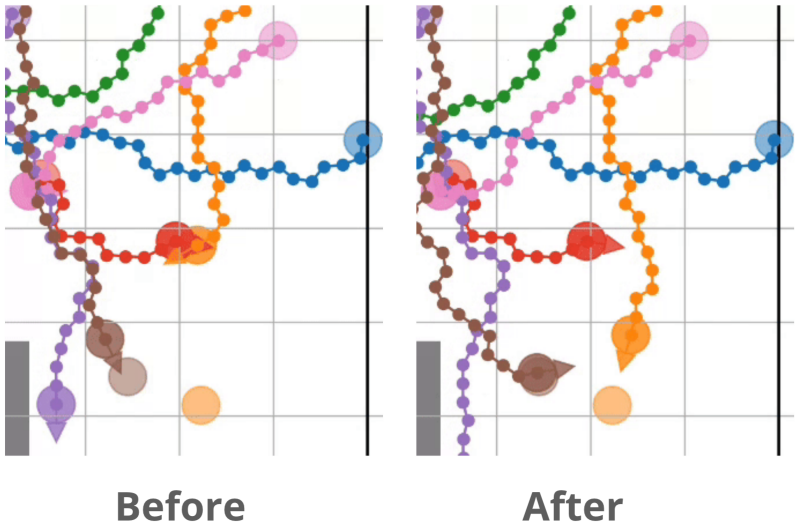

Fig. 3. Comparison of robot trajectories before (left) and after (right) applying CBS. The right panel demonstrates collision-free paths achieved by resolving conflicts dynamically where as the left shows the red and yellow robot colliding.

This layer effectively combines global trajectory planning for the leader with local adjustments for the followers, ensuring smooth and cohesive swarm behavior in dynamic environments.

IV. Results

The effectiveness of the conflict-based search layer for static goal state for each robot is demonstrated in Figure 3. The left panel shows the trajectories of robots before applying CBS, where several collisions occur due to overlapping paths of different robots. In contrast, the right panel illustrates the improved trajectories after applying CBS, where the robots maintain collision-free paths while navigating toward their goals. This result highlights the capability of CBS to detect and resolve conflicts by generating constraints and replanning individual robot paths to ensure safe coordination within the swarm.

The results presented in Figure 2 demonstrate the performance of the hierarchical framework in a scenario where a leader follows a predefined trajectory while maintaining a specific swarm formation. At timestep $t = 80.0$, a formation change is introduced, requiring the swarm to adapt dynamically to achieve the new desired configuration. By timestep $t = 167.0$, the robots successfully rearrange into the desired formation while ensuring collision-free navigation.

- **Scenario Description:** The swarm starts in an initial formation, following a predefined trajectory led by the leader. At a predefined timestep ($t = 80.0$), the formation changes, testing the system's ability to handle dynamic transitions.
- **Dynamic Adaptation and Formation Control:** The robots dynamically adapt their positions to meet the updated formation requirements while following the leader. By timestep $t = 167.0$, the desired formation is successfully achieved, as evident from the figure.
- **Self-Collision and External Collision Avoidance:** The framework ensures that the robots avoid both self-collisions within the swarm and collisions with environmental obstacles (represented by gray regions). The paths taken by the robots demonstrate the effective integration of collision-free path planning and conflict resolution.

- **Sequential Progression:** The sequential snapshots in Figure 2 highlight the gradual transition of the swarm as it moves through the environment and responds to the change in formation. Each timestep reflects coordinated motion and adherence to formation goals.
- **Leader-Follower Model Validation:** The virtual leader robot successfully guides the swarm through the environment, and the follower robots adjust their paths accordingly to maintain the overall formation integrity.

The test scenario validates the robustness of the proposed framework in managing dynamic formation changes, obstacle avoidance, and seamless coordination within a swarm of robots.

V. Conclusion

This paper introduced a hierarchical framework for adaptive path planning in swarm robots operating under a leader-follower model. By integrating Conflict-Based Search (CBS) for conflict resolution and motion primitive-based Rapidly-exploring Random Tree (RRT) for individual robot navigation, the system effectively manages dynamic formation control, collision-free navigation, and real-time conflict resolution. The results demonstrate the framework's ability to handle dynamic formation changes and environmental obstacles, ensuring coordinated swarm behavior.

Future work can focus on optimizing the computational efficiency of the framework, as the current implementation requires 5-6 minutes for planning in the tested scenarios and scales significantly with larger swarms or more complex environments. Achieving real-time implementation would be a crucial advancement, enabling deployment in dynamic, real-world applications.

Acknowledgment

The authors would like to thank Prof. Maxim Likachev for his guidance throughout this project.

References

[1] S. Javed, A. Hassan, R. Ahmad, W. Ahmed, R. Ahmed, A. Saadat, and M. Guizani, "State-of-the-Art and Future Research Challenges in UAV Swarms," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19023–19045, 2024. doi: 10.1109/JIOT.2024.3364230.

[2] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015. doi: 10.1016/j.artint.2014.11.006.

[3] X. Xu and Y. Diaz-Mercado, "Swarm herding: A leader-follower framework for multi-robot navigation," *arXiv preprint*, vol. arXiv:2101.07416, 2021. [Online]. Available: https://arxiv.org/abs/2101.07416.

[4] S. Javed, A. Hassan, R. Ahmad, W. Ahmed, R. Ahmed, A. Saadat, and M. Guizani, "State-of-the-Art and Future Research Challenges in UAV Swarms," *IEEE Internet of Things Journal*, vol. 11, no. 11, pp. 19023–19045, 2024. doi: 10.1109/JIOT.2024.3364230.

[5] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006. doi: 10.1109/JPROC.2006.876939.

[6] X. Li, Y. Wang, and J. Zhang, "Multi-Agent Reinforcement Learning for Collision-Free Navigation in Swarm Robotics," *Robotics and Autonomous Systems*, vol. 152, pp. 103872, 2022. doi: 10.1016/j.robot.2022.103872.

[7] T. Zhou, L. Chen, and F. Lin, "Heuristic Evolutionary Algorithms for Real-Time Motion Planning in High-Dimensional Swarm Systems," *IEEE Transactions on Cybernetics*, vol. 53, no. 2, pp. 1436–1447, 2023. doi: 10.1109/TCYB.2023.3156145.