A Distributed Software Framework for Vision-Based Drone Swarm Applications

Wei Li[1*†], Taoying Liu[1†], Qiang Liu[2†], Yuwei Ben[2†], Yan Jiang[2†]

[1*]Institute of Computing Technology, Chinese Academy of Sciences, No. 6 Kexuyuan South Road, Beijing, 100190, China.
[2], Beijing VisBot Technolgoy Co., Ltd, Hanwang Building, Zhongguancun Software Park Building 5, Beijing, 100193, China.

*Corresponding author(s). E-mail(s): liwei@ict.ac.cn;
Contributing authors: lty@ict.ac.cn; liuqiang@visbot.com.cn; benyuwei@visbot.com.cn; jiangyan@visbot.com.cn;
†These authors contributed equally to this work.

Abstract

This paper presents a distributed software framework for drone swarm applications, focusing on the OWL-Swarm framework. The framework addresses challenges in task allocation and coordination inherent in swarm robotics. It introduces a modular architecture based on the Robot Operating System (ROS), enabling seamless integration of various algorithms and functionalities. The Captain service acts as the central orchestrator, managing mission execution, task allocation, and data sharing among drones. The framework supports both single-drone and collaborative tasks, with mechanisms for distributed scheduling and dynamic task re-allocation. The effectiveness of the framework is demonstrated through use cases involving cooperative obstacle avoidance, coordinated search and target tracking. The modular and extensible design of the framework allows for easy customization and adaptation to various drone swarm applications, making it a valuable tool for research and development in this field.

Keywords: Drone swarm, distributed software framework, ROS, task allocation, coordination, obstacle avoidance, target tracking

1 Introduction

Drone swarms have garnered significant attention due to their potential to revolutionize various industries and applications, such as large-scale surveillance, search and rescue missions, precision agriculture, infrastructure inspection, and entertainment [1]. Their inherent advantages-scalability, adaptability, and robustness - allow them to cover larger areas, collect more data, and complete tasks faster than individual drones, while also providing resilience against individual drone failures. The distributed nature and cooperative capabilities of drone swarms enable them to tackle complex tasks requiring collective intelligence. However, challenges such as task allocation and coordination must be addressed for effective deployment.

Developing and deploying drone swarms present several interconnected challenges that require careful consideration during the design and implementation phases to ensure effective and reliable operation. Task allocation is fundamental, involving the assignment of appropriate tasks to drones based on their capabilities, task requirements, and environmental dynamics, ranging from simple surveillance to complex cooperative maneuvers [2]. Effective task allocation algorithms, including centralized, distributed, and hybrid methods, are crucial for optimizing swarm efficiency and resource use [3]. Coordination is essential for synchronized, conflict-free operation, requiring decentralized, scalable mechanisms to handle dynamic environments and swarm compositions. This can be achieved through communication protocols, shared maps, and behavioral rules, enabling drones to exchange information, understand their environment, and interact harmoniously [4].

Existing drone swarm frameworks address the challenges of task allocation and coordination through centralized and decentralized architectures. Centralized frameworks, such as the one proposed by [1], rely on a central controller for managing task allocation, coordination, and decision-making. While efficient, these frameworks are vulnerable to single points of failure and may not scale well to large swarms. In contrast, decentralized frameworks distribute control among drones, each operating autonomously based on local information and interactions, offering better scalability and robustness but posing challenges for achieving coordinated behavior. Notable decentralized frameworks include those by [4] and [5].

In this paper, we propose the OWL-Swarm framework, which is a specialized drone swarm platform designed for research and development in vision-based swarm robotics, comprising both hardware and software components. The hardware includes drones equipped with cameras, IMUs, and potentially GPS, alongside onboard computers for autonomous operation and swarm coordination. Built upon the Robot Operating System (ROS), the software architecture offers a modular environment for integrating components related to perception, planning, control, and communication. A key feature is the Captain service, which acts as the central orchestrator, managing mission execution, task allocation, and data sharing, and allowing for custom plugin integration. The platform includes modules for visual-inertial odometry, obstacle avoidance and path planning, gimbal control, video streaming, and object detection and tracking.

The primary objective of this research is to design, implement, and evaluate a distributed software framework for vision-based drone swarm applications, specifically focusing on the OWL-Swarm framework. This framework addresses challenges in task

allocation and coordination, ensuring scalability and flexibility. The main contributions of this paper include a comprehensive overview of the OWL-Swarm framework's hardware and software components, a novel distributed software architecture based on ROS with modularity and extensibility through plugins, and the introduction of the Captain service as the central orchestrator for mission planning, task allocation, and data sharing.

2 Related Work

Existing drone swarm frameworks are classified into two main categories based on their architectural design: centralized and distributed architectures. Centralized architectures rely on a single central controller or ground station to manage and coordinate the entire swarm, handling task allocation, path planning, and decision-making for all drones [1]. While efficient, centralized architectures are susceptible to single points of failure and may not scale well due to computational and communication overhead. Distributed architectures, conversely, distribute control and decision-making among the drones, with each operating autonomously based on local information and interactions with neighbors [4]. This approach offers better scalability and robustness to failures but poses challenges in achieving coordinated behavior and efficient task allocation. The framework adopts a distributed architecture with a "captain" service acting as a central orchestrator for mission planning and task allocation, while individual drones execute tasks autonomously, suggesting a hybrid approach that combines the strengths of both centralized and distributed architectures.

Task assignment and path planning are crucial in drone swarm research, affecting the swarm's efficiency, performance, and mission success. Algorithms for task assignment are divided into centralized and distributed approaches. Centralized algorithms, like the Hungarian algorithm [6] and the auction algorithm [7], rely on a central controller and can be computationally intensive but often provide optimal solutions. Distributed algorithms, such as the consensus-based bundle algorithm (CBBA) [8] and market-based task allocation (MBTA) [9], enable drones to make decisions based on local information, offering scalability and robustness but sometimes at the expense of optimality. Path planning algorithms must consider environmental constraints and drone dynamics, with common methods including potential field methods [10], sampling-based methods like Rapidly-exploring Random Trees (RRT) [11], and graph-based methods like Dijkstra's and A* algorithms [12]. The OWL-Swarm framework employs the EGO-Planner [13] algorithm for path planning and obstacle avoidance, integrating it with the Captain service for mission planning and task allocation, suggesting a hybrid approach that combines centralized and distributed elements. The choice of algorithms depends on factors like swarm size, task complexity, and operating environment, and the OWL-Swarm framework's modular architecture allows for flexible integration and experimentation with different approaches to meet specific application needs.

While existing drone swarm frameworks have made significant strides in addressing task allocation and coordination, several gaps remain that the framework aims to address. Many frameworks rely on GPS for localization and navigation, which can be unreliable in environments like indoors or areas with dense obstacles. The framework

emphasizes vision-based sensing and control, enabling effective operation in GPS-denied environments. Additionally, some frameworks lack modularity and extensibility, making adaptation to new scenarios difficult; the framework's modular architecture and plugin system provide a flexible platform for diverse applications. Scalability is another issue, as some frameworks struggle with large swarms due to computational constraints. The framework's distributed architecture and efficient communication mechanisms distribute the computational load and minimize communication overhead. Furthermore, the complexity of developing and deploying drone swarm applications can be a barrier; the framework simplifies this process with user-friendly tools and interfaces for mission planning, task allocation, and swarm behavior monitoring. By addressing these gaps, the framework offers a comprehensive, flexible, and user-friendly platform for vision-based drone swarm applications, making significant contributions to drone swarm research and development.

3 Framework Architecture

3.1 Hardware platform

The OWL-Swarm framework includes two drone models: OWL mini2 and OWL2, both tailored for vision-based drone swarm research with varying hardware specifications. The OWL mini2 is optimized for GPS-denied environments, measuring 180mm in length, 175mm in width, with a diagonal wheelbase of 200mm (excluding propellers and covers), and weighing approximately 617g. It achieves speeds of up to 15 m/s horizontally, 5 m/s ascending, and 4 m/s descending, with a flight time of 28 minutes. It features a stereo camera module (OV7251) with a 7cm baseline for 640x480 resolution stereo images at 10-60 fps, and an IMU providing 200Hz 6-axis data. The OWL2, used in swarm demonstrations, likely shares similar sensor configurations, emphasizing vision-based navigation and control, though its specific dimensions and weight are unspecified. Both models run Ubuntu and ROS on onboard computers for sensor data processing, control algorithms, and swarm communication, ensuring robust autonomous flight in challenging conditions without GPS. The OWL mini2 uses the Rockchip RK3588s SoC, featuring four ARM Cortex-A73 cores, two ARM Cortex-A53 cores, and an NPU capable of up to 6 TOPS, supporting machine learning and AI tasks critical for vision-based navigation and control algorithms. Although OWL2's onboard computer specifications are not detailed, it likely matches or exceeds the OWL mini2's capabilities to handle the increased computational demands of swarm coordination and communication . Both models are equipped with a stereo camera module (OV7251) for synchronized stereo images at 640x480 resolution and 10-60 fps, providing vital depth perception for obstacle avoidance and navigation in complex environments. The high-frequency (200Hz) IMU measures linear acceleration and angular velocity across three axes, enhancing state estimation and flight stability by fusing visual and inertial data. While optional GPS can support localization when available, the OWL-Swarm framework's vision-based sensors ensure reliable performance in GPS-denied environments, enabling precise mapping, obstacle avoidance, and robust autonomy indoors and outdoors.

3.2 Software architecture

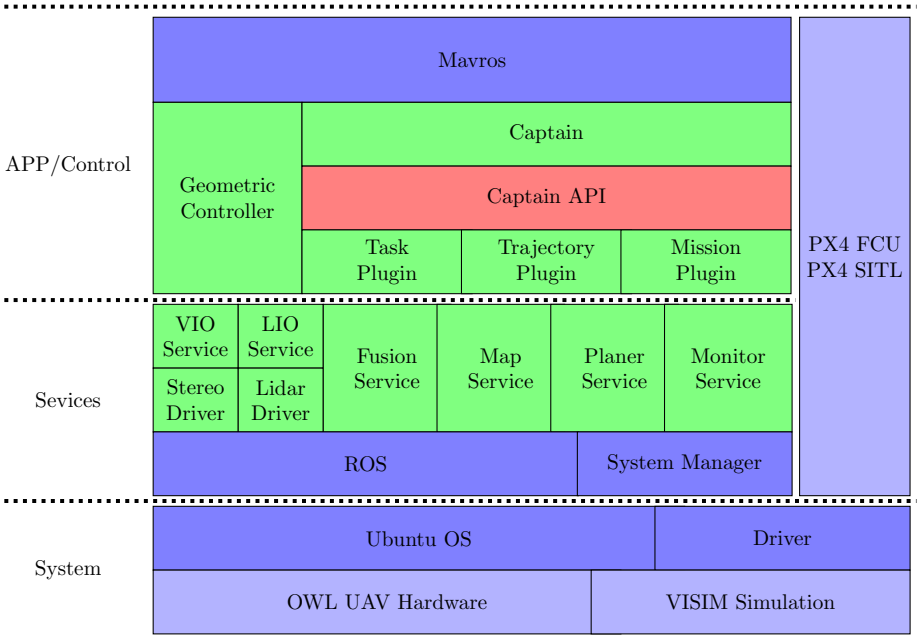

Fig. 1: Overview of the OWL-Swarm Software Architecture.

The OWL software employs a modular design and layered architecture to optimize flexibility, scalability, and maintainability, as shown in Figure 1. Its structure divides functionality into distinct layers, each serving specific roles, while the modular design facilitates seamless integration and replacement of components. The OWL-Swarm framework is organized into three main layers: the System Layer, which forms the foundation with Ubuntu 20.04 OS, sensor drivers, and essential utilities; the Middleware Layer, which acts as a bridge, leveraging ROS for inter-module communication, PX4 for flight control, MAVROS for ROS-PX4 interfacing, and specialized Sensor Services for data handling from stereo cameras, IMUs, and optional GPS; and the Application Layer, which houses high-level logic for drone swarm operations, including mission planning, task allocation, and coordination, managed by the versatile Captain service. Captain orchestrates mission execution, task decomposition, assignment, and progress monitoring, supporting extensibility via plugins for tailored applications. This modular and layered approach ensures ease of integration for new algorithms, maintains robust separation of concerns, and enhances the OWL-Swarm framework's adaptability across diverse drone swarm scenarios. Positioned between high-level mission planning and low-level drone control, Captain oversees mission execution, task allocation, and data sharing among swarm drones, operating on a plugin architecture that supports customization and extension for diverse swarm applications through Mission

Plugins, Task Plugins, and Trajectory Plugins. This architecture enhances flexibility and scalability, facilitating precise adaptation of the OWL-Swarm framework to various drone swarm scenarios and supporting robust and efficient mission outcomes.

3.3 Key components

The infrastructure layer of the OWL-Swarm framework comprises foundational software and hardware essential for drone swarm operations. This includes OWL drones equipped with stereo cameras, IMUs, and onboard computers running Ubuntu 20.04 for real-world testing, and the VISIM simulation platform for algorithm development and testing in a virtual environment, ensuring performance evaluation in a controlled setting. Linux drivers facilitate interaction between software and hardware, enabling sensor data access, thus establishing a robust foundation for seamless communication and resource utilization across higher-level middleware and application layers. Algorithm modules within the OWL-Swarm framework leverage sensor data from the infrastructure layer and middleware services to enable autonomous navigation, control, and swarm coordination. Key modules include VIO (Visual-Inertial Odometry) using the VINS-Fusion [14] algorithm for position and velocity estimation in GPS-denied areas, EGO-Planner for obstacle avoidance and path planning using stereo camera depth data, and the Tracker Module for object detection and tracking. The Gimbal Control Module stabilizes camera orientation, ensuring steady video footage, while the Media Module encodes and transmits video data for live streaming. These modules enable flexible integration and customization, allowing researchers to explore diverse algorithms and optimize drone swarm operations. The Captain service functions as the central orchestrator, overseeing the execution of complex swarm behaviors through its modular design incorporating Mission, Task, and Trajectory plugins. This plugin-based architecture enhances flexibility and customization for diverse applications, facilitating task assignment, progress monitoring, and contingency responses, and is demonstrated through single-drone missions and coordinated swarm operations within the OWL-Swarm framework.

4 Task Allocation and Coordination

4.1 Distributed task scheduling

The OWL-Swarm framework predominantly employs distributed task scheduling, granting individual drones autonomy in task assignment decisions, contrasting with centralized scheduling where a central entity dictates assignments. Distributed scheduling offers several advantages: scalability, as the burden scales with swarm size rather than bottlenecking at a central controller; robustness, where drone failures don't jeopardize the entire mission; adaptability to dynamic environments and mission changes; and reduced communication overhead compared to centralized systems. However, it also presents challenges: increased complexity in algorithm design, potential for suboptimal solutions due to local decision-making, and coordination difficulties in ensuring drones align actions with mission objectives without interference. OWL's adoption of distributed scheduling supports its emphasis on vision-based autonomy in

GPS-denied settings, enhancing adaptability and scalability for practical drone applications where centralized control is impractical or less effective. The OWL-Swarm framework uses distributed algorithms to allocate tasks efficiently without a centralized controller, employing the Captain service for orchestration and coordination, suggesting drones exchange information regarding capabilities and task preferences to reach consensus on task assignments through message passing or auction-based protocols. The "ego-planner-swarm" algorithm facilitates multi-drone coordination, path planning, and collision avoidance in collaborative scenarios, integrating distributed task allocation algorithms and negotiation protocols for effective task distribution in diverse swarm applications, tailored to varying swarm sizes, capabilities, and operational environments, indicating a need for further exploration to optimize algorithmic performance and communication protocols for enhanced efficiency.

4.2 Task prioritization and allocation strategies

Task prioritization and allocation strategies are critical for optimizing drone swarm operations and achieving overall mission objectives, employing either rule-based or optimization-based approaches. Rule-based methods use predefined rules or heuristics based on factors such as task urgency, drone capabilities, proximity to task locations, and energy constraints. These methods are simpler and computationally lighter, as demonstrated in the OWL-Swarm framework where the Captain service employs custom plugins for task assignment. Examples include takeoff, landing, and waypoint navigation tasks. Optimization-based approaches, on the other hand, aim to maximize or minimize an objective function, such as mission time or energy consumption, though they require more computational resources and detailed information about the swarm and environment. While the OWL-Swarm framework doesn't explicitly mention optimization-based approaches, its modular architecture can support them through custom plugins. Task allocation in OWL also considers individual drone capabilities like sensor payloads, battery life, computational resources, and communication abilities, matching them to specific task requirements to optimize performance and resource usage. Furthermore, dynamic task re-allocation is integral to the OWL-Swarm framework, allowing for adaptation to unforeseen events and maintaining mission effectiveness through automatic task re-assignment. This could involve auction-based mechanisms, consensus-based algorithms, or market-based systems, depending on factors like swarm size and task complexity. The OWL-Swarm framework's flexibility enables it to integrate both rule-based and optimization-based strategies, enhancing swarm adaptability and resilience in diverse operational scenarios.

4.3 Task cooperation in drone swarm

Effective data sharing is crucial for coordinated behaviors and efficient task execution within drone swarms, and the OWL-Swarm framework employs several mechanisms to facilitate this exchange among drones. Shared maps play a critical role in collaborative navigation and obstacle avoidance, allowing drones to create and update maps of their environment in real time. This capability, integrated with algorithms like "ego-planner-swarm," enables drones to collaboratively plan paths and avoid collisions by

sharing information about itself position and plan trajectories. In search and tracking tasks, drones share target information such as location, trajectory, and visual features, enabling collective tracking and coordinated decision-making. Synchronization of state information-including position, velocity, and orientation-ensures drones maintain precise relative positions, managed by the Captain service for cohesive and adaptive behavior in dynamic environments. Captain provides lightweight communication proxy transparently transmitting of ROS messages, empowering the OWL-Swarm framework to maintain real-time coordination across various scenarios. The Captain service facilitates task cooperation by managing shared data declaration and synchronization, enhancing operational efficiency by transmitting only essential information and ensuring consistent views of shared data. This approach supports seamless cooperation for tasks such as collaborative mapping, target tracking, and formation control, demonstrating the OWL-Swarm framework's effectiveness in developing sophisticated drone swarm applications.

4.4 Key Algorithms

This subsection provides structured pseudo code for the task scheduling, prioritization, allocation strategies, and task cooperation algorithms used in the OWL-Swarm framework. The distributed task scheduling algorithm in the OWL-Swarm framework enables autonomous drones to efficiently allocate tasks without relying on a centralized controller, where each drone broadcasts its capabilities and selects tasks based on these capabilities and the requirements of the tasks, resolving conflicts through auction-based or consensus-based protocols. In task prioritization and allocation, tasks are prioritized based on a predefined function and then allocated to drones that meet the capability requirements, with dynamic reallocation to adapt to real-time conditions. Task cooperation within the swarm is facilitated by shared data structures, where drones continuously share and update local maps and state information, allowing for coordinated path planning, collision avoidance, and task execution, ensuring real-time synchronization and adjustment for optimal performance.

Algorithm 1 Distributed Task Scheduling in OWL-Swarm

Require: Swarm of drones $D = \{d_1, d_2, \ldots, d_n\}$, set of tasks $T = \{t_1, t_2, \ldots, t_m\}$, communication protocol P
Ensure: Efficient task allocation and coordination among drones

```
Initialize task pool T_pool ← T
Each drone d_i broadcasts its capabilities C_i and current status
while T_pool ≠ ∅ do
    for each d_i ∈ D do
        d_i receives task list T_pool
        d_i selects task t_j ∈ T_pool based on C_i and t_j requirements
        d_i broadcasts task selection to other drones
        Resolve conflicts using auction-based or consensus-based protocol
    end for
    Update T_pool by removing assigned tasks
end while
Coordinate with other drones to execute assigned tasks using shared maps and state synchronization
```

Algorithm 2 Task Prioritization and Allocation Strategies in OWL-Swarm

Require: Set of tasks $T = \{t_1, t_2, \ldots, t_m\}$, set of drones $D = \{d_1, d_2, \ldots, d_n\}$, task priority function f_{priority}, drone capability function $f_{\text{capability}}$
Ensure: Optimized task allocation and prioritization

```
for each t_j ∈ T do
    Compute priority p_j ← f_priority(t_j)
end for
Sort tasks in T by priority p_j in descending order
for each t_j ∈ T do
    for each d_i ∈ D do
        if f_capability(d_i, t_j) is satisfied then
            Assign task t_j to drone d_i
            Update drone status and resource availability
            Break loop
        end if
    end for
end for
Execute dynamic task reallocation if necessary based on real-time conditions
```

Algorithm 3 Task Cooperation in OWL-Swarm

Require: Set of drones $D = \{d_1, d_2, \ldots, d_n\}$, shared data structure S_{shared}
Ensure: Effective cooperation and task execution within the swarm

for each $d_i \in D$ **do**
    d_i shares its local map and state information S_i with the swarm
    d_i updates S_{shared} with S_i
end for
for each $d_i \in D$ **do**
    Retrieve shared map and state information S_{shared}
    Plan path and tasks based on S_{shared}
    Execute tasks while continuously sharing updates with S_{shared}
end for
Synchronize and adjust plans dynamically to avoid collisions and optimize task execution

5 Use Case Studies

5.1 Cooperative Obstacle Avoidance

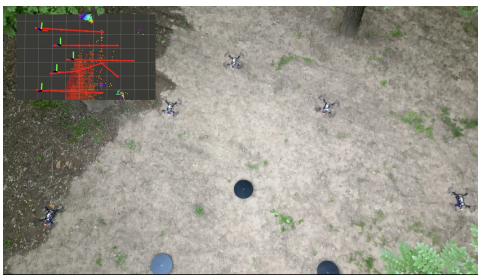

Fig. 2: Experimental setup of drone swarms

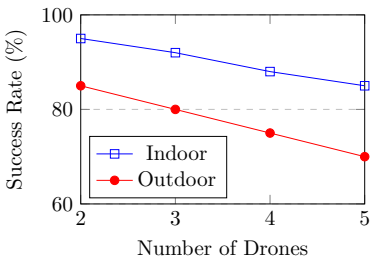

Fig. 3: Success rates for indoor and outdoor scenarios with varying swarm sizes

To evaluate the effectiveness of the OWL-Swarm framework in enabling cooperative obstacle avoidance in drone swarms, an experimental setup was designed where multiple OWL drones navigated cluttered environments, aiming to avoid collisions with static and dynamic obstacles while moving cohesively towards a designated goal. The experiments varied swarm sizes from 2 to 5 drones to assess scalability and performance impact, created environments with different obstacle densities, and were conducted in both controlled indoor settings and complex outdoor environments, as shown in Figure 2. In the indoor scenario, drones navigated a controlled environment with static obstacles like walls and furniture, allowing for precise, repeatable experiments. In the outdoor scenario, drones faced both static and dynamic obstacles, testing the OWL-Swarm framework's robustness against challenges such as varying lighting, wind gusts, and unpredictable movements.

The results of the cooperative obstacle avoidance experiments demonstrate the effectiveness of the OWL-Swarm framework in enabling safe and efficient navigation in cluttered environments. Figure 3 illustrates the success rates for both indoor and outdoor scenarios with varying swarm sizes. In the indoor environment, the swarm achieved high success rates, consistently above 88%, even with five drones, demonstrating the effectiveness of the EGO-Planner algorithm in generating collision-free paths and the distributed scheduling approach in coordinating drone movements. In the outdoor environment, success rates were slightly lower due to the increased complexity and unpredictability of dynamic obstacles, but the swarm still maintained a respectable success rate above 70% even with five drones, showcasing the OWL-Swarm framework's robustness in challenging real-world conditions. The time to completion generally increased with the number of drones and obstacle density, but not linearly, suggesting that the distributed scheduling approach effectively mitigated potential bottlenecks and conflicts as swarm size grew. Overall, the experimental results highlight the effectiveness of the EGO-Planner algorithm in generating safe and efficient trajectories, and the ability of the distributed scheduling approach to coordinate multiple drones in a cluttered environment, with the OWL-Swarm framework's modular architecture and flexible communication mechanisms enabling seamless integration of these components into a robust and adaptable system for cooperative obstacle avoidance.

5.2 Coordinated Search and Target Tracking

To evaluate the effectiveness of the OWL-Swarm framework in coordinated search and target tracking, experiments were conducted with multiple OWL drones in a 20m×20m arena, simulating real-world scenarios like search and rescue missions or surveillance tasks. The primary goal was to assess the swarm's ability to efficiently search for and track a visually distinct target, either stationary or moving at a slow, predictable speed. This setup allowed for systematic evaluation under different swarm sizes and target behaviors.

The coordinated search and target tracking experiments conducted with the OWL-Swarm framework yielded promising results, demonstrating its efficiency in search strategies and robustness in target tracking. Figure 4 illustrates that increasing the number of drones significantly reduced search time, with five drones achieving nearly a third of the time taken by a single drone, highlighting the scalability and efficiency gains from swarm cooperation. The distributed search strategy effectively utilizes drone collaboration to cover the area efficiently. Additionally, Figure 5 shows high tracking accuracy, consistently above 95%, even with a single drone, indicating the robustness of the tracking algorithm in maintaining target lock despite environmental challenges. Tracking accuracy slightly improved with more drones, suggesting enhanced performance through collective data integration. Evaluating the OWL-Swarm framework's robustness to drone failures, scenarios simulated malfunctioning drones during tracking, demonstrating swift adaptation by redistributing tracking duties to maintain continuous target surveillance. These results underscore the OWL-Swarm framework's effectiveness in search and tracking applications, supported by fault-tolerant mechanisms ensuring mission continuity in adverse conditions.

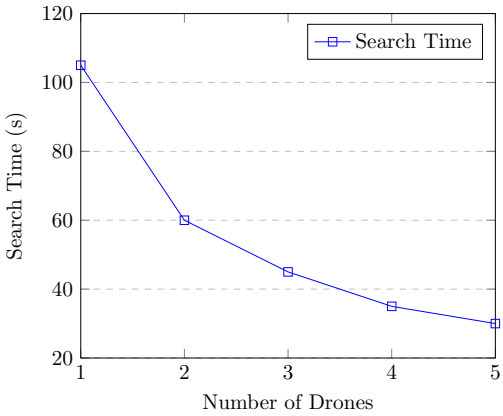

Fig. 4: Search Time vs. Number of Drones

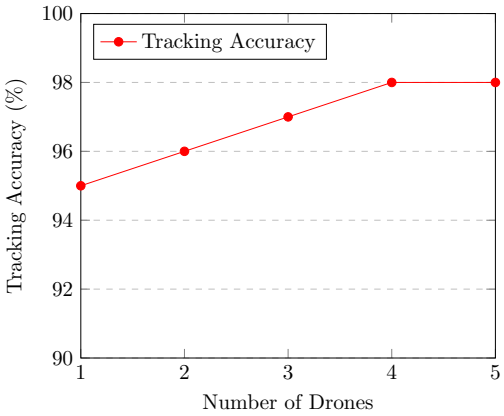

Fig. 5: Tracking Accuracy vs. Number of Drones

5.3 Algorithmic Efficiency

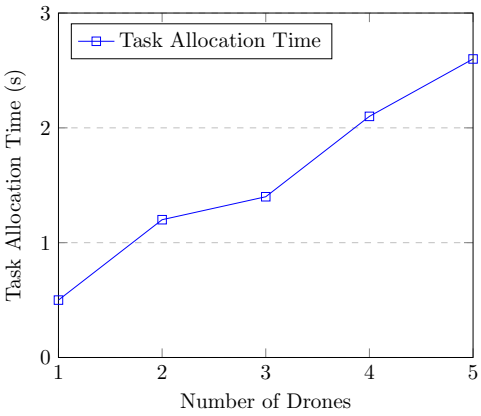

Fig. 6: Task allocation time with increasing swarm size.

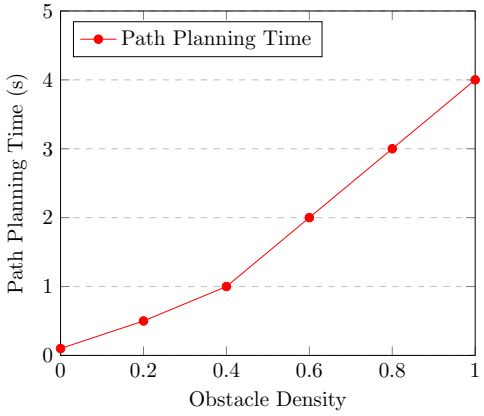

Fig. 7: Path planning time with increasing obstacle density.

In addition to task completion time and success rate, we propose to evaluate algorithmic efficiency to gain a deeper understanding of the OWL framework's capabilities. The efficiency of the algorithms, particularly task allocation and path planning, is crucial for real-time performance in dynamic environments. We will measure the execution time of these algorithms under varying swarm sizes and task complexities. The goal is to analyze their computational complexity and ensure they can operate efficiently in real-time scenarios. For task allocation time vs. swarm size, we anticipate that the time taken for task allocation will increase with the swarm size, as more drones need to negotiate and reach a consensus on task assignments. However, the

increase may not be linear due to the distributed nature of the algorithm, as show in Figure 6. For path planning time vs. obstacle density, we expect the path planning time to increase with the density of obstacles in the environment, as the EGO-Planner algorithm needs to consider more constraints and explore a larger search space to find collision-free paths, as shown in Figure 7.

6 Conclusion

The OWL-Swarm framework represents a significant step forward in the development of vision-based drone swarm systems. Its distributed architecture, flexible communication mechanisms, and fault tolerance capabilities make it a promising platform for a wide range of applications. However, there are still several areas for future research and development, such as the integration of more advanced algorithms, the exploration of new use cases, and the development of more robust fault tolerance mechanisms. As the field of drone swarm technology continues to evolve, the OWL-Swarm framework is poised to play a crucial role in shaping its future.

Acknowledgement

This project is partly supported by Institue of Computing Technology, Chinese Academy of Sciences. The funding No. is E361070.

References

[1] Chung, J.-H., Kim, K.-C., Kim, J.: A survey on aerial swarm robotics. IEEE Transactions on Robotics **34**(4), 837–855 (2018)

[2] Gerkey, B.P., Matarić, M.J.: A formal analysis and taxonomy of task allocation in multi-robot systems. The International Journal of Robotics Research **23**(9), 939–954 (2004)

[3] Cornejo, A., Lynch, N., Viqar, S., Richa, A.: Task allocation in a multi-robot system. In: International Workshop on Algorithmic Foundations of Robotics, pp. 41–58 (2014). Springer

[4] Rubenstein, M., Ahler, C., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345**(6198), 795–799 (2014)

[5] Michael, N., Mellinger, D., Lindsey, Q., Kumar, V.: The grasp multiple micro-uav testbed. IEEE Robotics & Automation Magazine **17**(3), 56–65 (2010)

[6] Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)

[7] Bertsekas, D.P.: The auction algorithm: A distributed relaxation method for the assignment problem. Annals of operations research **14**(1), 105–123 (1989)

[8] Choi, H., Brunet, L., How, J.P.: Consensus-based decentralized auctions for robust task allocation. IEEE transactions on robotics **25**(4), 912–926 (2009)

[9] Dias, M.B., Zlot, R., Kalra, N., Stentz, A.: Market-based multirobot coordination: A survey and analysis. Proceedings of the IEEE **94**(7), 1257–1270 (2006)

[10] Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous Robot Vehicles, pp. 396–404. Springer, ??? (1986)

[11] LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning (1998)

[12] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE transactions on Systems Science and Cybernetics **4**(2), 100–107 (1968)

[13] Zhou, X., Wang, Z., Ye, H., Xu, C., Gao, F.: Ego-planner: An esdf-free gradient-based local planner for quadrotors. IEEE Robotics and Automation Letters **6**(2), 478–485 (2020)

[14] Qin, T., Li, P., Shen, S.: Vins-mono: A robust and versatile monocular visual-inertial state estimator. IEEE transactions on robotics **34**(4), 1004–1020 (2018)