A Swarm Algorithm for Following Formations Applied to Crazyflie Drones

Oliver Kosak*, Philipp Kastenmüller, Vinzenz Malke, Fabian Schwaiger and Wolfgang Reif

Institute for Software and Systems Engineering at the University of Augsburg, 86159, Universitätsstraße 6a, Germany

Abstract

The control of drone swarms and their targeted application for various use cases, such as search and rescue missions, monitoring of critical infrastructure, or agriculture, is a current research field. In line with this, we present in this paper a new swarm mechanism that enables drone swarms to move collectively in chain-like formations in a self-organizing manner providing an alternative to centrally controlled leader-following approaches. We demonstrate that the behavior enables the typical self-organizing capabilities: scalability when expanding the swarm size, robustness against failures, and flexibility in adapting to changing requirements. Therefore, we provide a prototypical implementation in simulation and conduct a proof-of-concept experiment using a swarm of real Crazyflie drones.

Keywords

swarm, drones, formation flight, following, self-organization

1. Introduction

Swarm intelligence, inspired by the collective behaviors observed in nature, has motivated scientific research for decades. The self-organizing principles of swarms enable remarkable properties such as robustness, scalability, and adaptability, which are highly beneficial for both natural and technical systems. In particular, swarm robotics with focus on drones has emerged as a promising field [1], leveraging these principles to address challenges in areas such as search and rescue, environmental monitoring, and formation flight [2, 3, 4, 5, 6]. Formation flight currently often relies on centralized control to synchronize movements of numerous drones. In this paper, we introduce a swarm-based algorithm for following formations in the spirit of behavior based formation control following the definitions of [7], classifying leader-follower as one of five "conventional" formation control methods. The proposed approach organizes drones into formation, guided by an externally controlled leader. Our swarm approach ensures adaptivity in dynamic environments, allowing the formation to recover from drone failures while also enabling the seamless integration of additional drones. We provide our prototyping environment, a simulation implemented in NetLogo (https://www.netlogo.org), readers can use to verify the algorithm's positive behavior and properties (robustness, scalability, flexibility), we depict in Figure 1b and whose full code be found on our GitHub repository (https://github.com/isse-augsburg/netlogo-following).To evaluate the effectiveness of our algorithm, we give proof-of-concept using a swarm of 7 Crazyflie 2.1 (https://www.bitcraze.io/products/crazyflie-2-1/) drones executing a ROS2 based verson of the swarm following in our indoor Vicon (https://www.vicon.com) flight arena. The infrastructure in that arena [8] allows for the positioning of individual drones, communication among neighboring drones, and the abstraction of many critical hardware control details, such as takeoff and landing maneuvers, and collision avoidance. We describe the experimental results in this paper and refer the reader to documenting video materials.

We first define the problem of dynamic following in drone swarms in section 2. In section 3, we detail our algorithm. In section 4, we present a proof-of-concept evaluation of our algorithm showcasing it in

8th International Workshop on Robotics Software Engineering (RoSE26)

*Corresponding author.

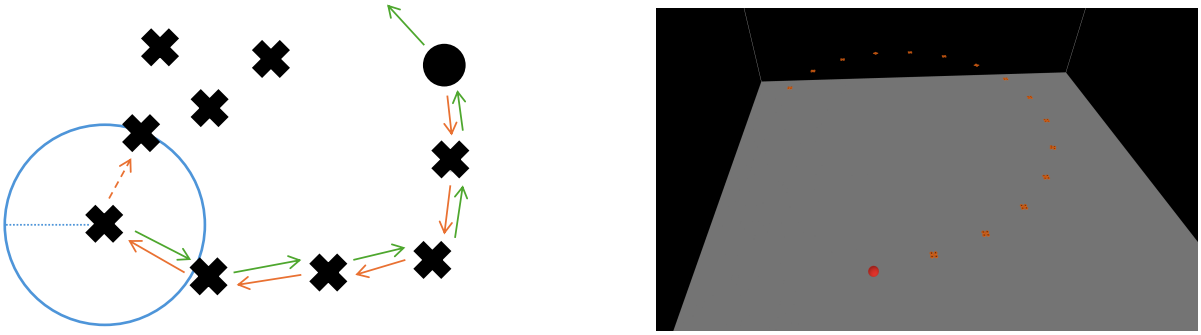
kosak@isse.de (O. Kosak); kastenmueller@isse.de (P. Kastenmüller); vinzenz.malke@uni-a.de (V. Malke); fabian.schwaiger@uni-a.de (F. Schwaiger); reif@isse.de (W. Reif)

0000-0003-0563-9797 (O. Kosak); 0009-0007-1342-5437 (P. Kastenmüller); 0009-0006-7482-7037 (V. Malke); 0009-0002-2718-4834 (F. Schwaiger); 0000-0002-4086-0043 (W. Reif)

a real-world scenario. We review related literature in section 5. Finally, we summarize our contributions and discuss potential directions for future research in section 6.

2. The Dynamic Following Problem

We define the dynamic following problem by adapting formulations for unstructured following from [9] for unmanned aerial vehicles or approaches for structured followgin from comparable problem domains, e.g., platooning [10]. Given a set of agents (drones) and a central moving leader (`ref`), the objective is to organize the agents into a chain-like formation. In this formation, each agent follows the one directly in front of it, thereby creating a chain. The formation must be dynamic, allowing for the addition and removal of agents while preserving the overall structure and movement toward a target destination. This problem represents an instance of the dynamic assignment problem where we want all agents in the system to join the formation denoted as `chain`. Initially only the central moving leader (`ref`) is part of the `chain`, and all other agents are unconnected. Each agent α_i in a set of agents $\mathbb{A}$ must be assigned to a leader $\alpha_j \in \mathbb{A} \cup \texttt{ref}$ such that α_i follows α_j to joint the `chain`. This relationship is denoted as $\texttt{follows}(\alpha_i, \alpha_j)$, where $\alpha_i \neq \alpha_j$. Additionally, all agents can follow only one leader and be followed by only one follower. This guarantees the formation of a directed graph that represents the leader-follower relationships. If only one agent remains without a follower, the chain is considered complete. To facilitate the description of relationships within the formation, we define the inverse of the `follows` predicate (i.e., $\texttt{follows}^{-1}$), as the `leader` predicate. An illustration of this concept is provided in Figure 1a. We extend the problem to include dynamic recovery capabilities, enabling the system to adapt to changes such as agent failures or removals. If any agent loses its follower (e.g., due to hardware failure or battery depletion), the assignments for that agent and all subsequent agents in the chain become invalid and must be reestablished. Therefore, we require a solution to the problem to be self-healing, meaning it can automatically recover from such disruptions by reassigning agents to new leaders and followers as necessary, ensuring the continuity of the formation.



(a) Illustration of the following problem.

(b) Solution to the problem in NetLogo

Figure 1: Following commanded by a central moving leader. In the conceptual illustration (a), the leader is symbolized as a dot. Orange arrows indicate follower agents (crosses), while green arrows indicate the leader. The neighborhood radius is represented by a blue circle. Dashed orange arrows denote potential follower relationships for agents without a leader. In the NetLogo simulation (b), the central moving leader is represented by a red dot, while the follower agents are represented as drone-pictograms.

3. Approach

Our approach to solving the dynamic following problem is based on a decentralized, self-organizing swarm algorithm. The key components of the algorithm include agent roles and relationships, a mechanism for recruiting followers, a movement strategy based on leader–follower dynamics, and a self-healing mechanism for dynamic recovery. The descriptions provided here are presented in

Algorithm 1 Leader Recruiting a Follower

```
if agent has no follower then
    scan neighbors without leader
    if available follower found then
        set follower ← chosen agent
        ask follower to set leader ← self
```

Algorithm 2 Agent Movement Decision

```
if agent has a leader then
    compute vector v⃗ ← direction to leader
    scale v⃗ based on distance and urgency
    move along v⃗
else
    apply random or idle motion
```

pseudocode style and are based on our simulation implemented in NetLogo which can be found on our GitHub repository (https://github.com/isse-augsburg/netlogo-following).

Agent Roles and Relationships Each *swarm agent* operates in a decentralized manner, with the ability to dynamically establish and maintain leader–follower relationships. These relationships are fundamental to the formation and organization of the swarm. Specifically, each agent can have a *leader*, which is the agent it follows, and a *follower*, which is an agent that follows it.

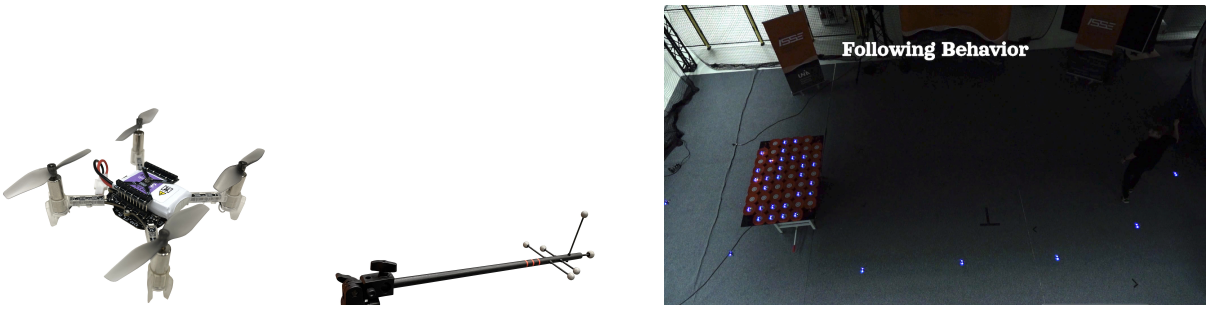
At initialization, agents are unconnected and operate independently. This means each agent starts with no `leader` or `follower` and sets up its internal state, including parameters such as current `position`, `movement vector`, and a `timer` for dynamic recovery. Over time, agents dynamically establish leader–follower links to form a chain. This process is driven by the presence of a *central moving leader* (e.g., a user controlled pointing device), which initiates the recruitment of followers.

Finding a Follower Leadership in the swarm is established in a top down manner, beginning with agents that are already connected, such as `ref` or agents in formation without followers. Initially only `ref` is connected, i.e., $\texttt{chain} = \{\texttt{ref}\}$. Agents $\alpha \in \texttt{chain}$ without a follower (i.e., the respective agent at the end of the chain) execute a localized search strategy to recruit a follower: Agents scan their immediate *neighborhood radius* to identify nearby agents that are not yet part of the swarm (i.e., agents without a leader). From the pool of unconnected agents, one is randomly selected to become the follower. Once a follower is successfully recruited the leader updates its internal state to record the new follower. The recruited follower updates its state to record the leader that selected it. This recruitment process is recursive, originating from `ref`. As each new follower is added to `chain`, it may itself become a leader, initiating its own recruitment process to propagate the chain further. This decentralized mechanism ensures scalability and robustness in the swarm's formation. The pseudocode in Algorithm 1 outlines this recruitment process.

Movement Strategy The movement strategy of each agent is governed by its leader–follower relationship, ensuring coordinated motion and cohesion within the swarm. The pseudocode in Algorithm 2 outlines the decision-making process for agent movement.

If an agent has a leader, it calculates a directional vector $\vec{v}$ pointing toward the leader's position. This vector is scaled based on the distance to the leader, with diminishing urgency as the agent approaches its leader. The scaled vector determines the agent's heading and forward movement, ensuring smooth and adaptive alignment with the leader's trajectory. Conversely, if an agent does not have a leader, it defaults to random or idle motion. This behavior allows unconnected agents to explore their surroundings, increasing the likelihood of being recruited by a leader during the swarm formation process.

Agent Removal and Recovery The algorithm exhibits *self-healing* capabilities, enabling it to reconfigure and maintain structural integrity following disruptions, such as the removal or failure of a leader. When an agent is removed (e.g. to simulate hardware failure), all agents previously connected to it lose their leader or follower references. This is achieved by using a timeout-based recovery mechanism (cf. Algorithm 3), where agents employ a decentralized mechanism to detect



(a) Crazyflie 2.1 drone (b) User interface wand (c) Chain-like formation using 7 drones.

Figure 2: Used hardware in our following experiment (c) with real drones (a) controlled with our magic wand (b).

and respond to disconnections. Each agent is equipped with a timer, which monitors the continuity of its leader–follower relationship. The timer is periodically *reset* upon receiving a "token" from its upstream leader, signifying the persistence of the connection. If the timer expires without being reset, the agent infers that its leader is no longer valid or connected. It *invalidates its leader and follower references*, effectively detaching itself from the swarm structure. The detached agent transitions into a state where it can be recruited by another leader, facilitating reattachment to the swarm. This dynamic recovery mechanism ensures that the swarm can autonomously adapt to structural changes, maintaining robustness and scalability in dynamic environments.

Algorithm 3 Agent Logic — Timeout-Based Recovery

```
1: if agent has a follower then
2:     pass token downstream to reset timer
3:     decrement timer-ticks
4:     if timer-ticks ≤ 0 then
5:         clear leader and follower links
```

4. Proof-of-Concept

We implemented the algorithm in a NetLogo simulation environment, which allows us for rapid prototyping and testing of swarm behaviors. There, we demonstrate the algorithm's ability to form a chain-like following formation, where each agent follows a leader, creating a dynamic and adaptable structure, which we document in video materials on our GitHub (https://github.com/isse-augsburg/netlogo-following). Users can remove agents from the system or add agents to the system, and the swarm will adapt accordingly. User input is possible with predefined scripts or using WASD-keyboard inputs for interactive control of the overall formation. In our Proof-Of-Concept experiment, a human operator controls the movement of the leader drone using WASD-keyboard inputs or by predefined scripts. After an initially random positioning, the drones begin to integrate into the existing formation as soon as the formation comes into their line of sight, until all drones are part of the chain-like formation. The drones successfully follow the leader as the operator moves it through the simulation environment and adjust their positions to maintain the formation. Following a stable phase, an externally triggered, controlled failure of a drone is induced, which leads to the decoupling of the subsequent drones. The remaining drones, no longer part of the formation, initially hover in place but quickly reintegrate into the formation. In the same way, new drones (green) integrate themselves after they are introduced.

Furthermore, we tested the algorithm with real drones, demonstrating its applicability in real-world scenarios. We replicate the experiment described in our Vicon flight arena using 7 out of 50 Crazyflie 2.1 drones (cf. fig. 2a), with position tracking and control implemented through a ROS2-based framework and documented it in video materials available in our YouTube playlist linked on GitHub

(https://github.com/isse-augsburg/netlogo-following).A human operator uses a physical pointing device (the *wand*, cf. fig. 2b) and collects drones hovering in our flight arena by walking close to them. The drones then follow the leader as the operator moves through the arena with the wand in its hand. Again, after a controlled failure of a drone (the drone lands), following drones decouple from the formation. The remaining drones, no longer part of the formation, initially hover in place but quickly reintegrate into the formation as soon as they are in range.

5. Related Work

Relevant literature we address in this paper includes studies on swarm behavior, formation flight, the queuing problem, and platooning as a related problem domain and case study. The queuing approach presented in [11] focuses on swarm behavior coordination with an emphasis on task allocation strategies. While their method demonstrates robustness and scalability in specific scenarios, it primarily addresses static environments in 2D and lacks the dynamic adaptability required for our application domain. In contrast, our approach emphasizes dynamic adaptability and real-time decision-making in complex, changing environments, making it more suitable for scenarios requiring continuous reconfiguration and responsiveness. The approach on vehicle formation forming outlined in [12] is tailored to automotive application domains, where the focus lies on specific, constrained scenarios. While it provides insights into task-specific implementations, it lacks the robustness and scalability required for broader, more dynamic environments. In contrast, our approach is designed to handle a wide range of scenarios with an emphasis on adaptability and scalability, making it more versatile for applications beyond the automotive domain. The survey on drone trajectory algorithms in [13] explores swarm behavior in drone systems, focusing on possible controlling strategies. However, it does not provide a detailed following algorithm. In contrast, our approach explicitly incorporates a robust following mechanism, enabling efficient adaptability in dynamic and complex environments, which is essential for real-world applications of drone swarms. The approach presented in [3] proposes a distributed method based on a virtual tube to address the formation exploration of micro multirotor unmanned aerial vehicles (UAVs). In contrast, our approach diverges by focusing on a more generalized framework that does not rely on predefined tube constraints or specific formation models. Instead, we emphasize adaptability and scalability in dynamic and unstructured environments, enabling real-time decision-making and coordination without the need for extensive prior knowledge or assumptions about the environment. This makes our method more versatile and applicable to a broader range of scenarios, including those requiring continuous reconfiguration and responsiveness in unpredictable conditions. In [14], the authors present a method for controlling a multi Crazyflie system for formation flight. Instead of local trajectory planning for each individual drone, following is the result of a global trajectory planning for the whole swarm.

6. Conclusion and Future Work

In this paper, we introduced a swarm algorithm for following behavior, which we apply to the formation flight of drones. We demonstrate the algorithm's ability to form a chain-like following formation, where each agent follows a leader, creating a dynamic and adaptable structure. We provide a prototype implementation of the algorithm in a NetLogo simulation environment, allowing for rapid prototyping and testing of swarm behaviors and migrated the prototype to a ROS2 implementation. Further, we evaluated the algorithm with a real swarm of Crazyflie 2.1 drones in an indoor flight arena. In a of proof-of-concept experiment, we demonstrate that this algorithm shows the key benefits of swarm algorithms, i.e., scalability, robustness and flexibility. Failures of agents are compensated by the self-organizing nature of the algorithm, ensuring that the formation remains stable despite unexpected events. Future work will focus on improving the algorithm's performance and robustness, as well as exploring additional applications of the following mechanism, performing empirical evaluation of its properties and integrating it in our swarm framework [15, 16, 17].

Declaration on Generative AI

During the preparation of this work, the authors used Copilot, Grammarly in order to: Grammar and spelling check, Paraphrase and reword. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Phadke, F. A. Medrano, C. N. Sekharan, T. Chu, An analysis of trends in uav swarm implementations in current research: simulation versus hardware, Drone Systems and Applications 12 (2024) 1–10. URL: https://doi.org/10.1139/dsa-2023-0099. doi:10.1139/dsa-2023-0099.

[2] M. Dorigo, G. Theraulaz, V. Trianni, Swarm robotics: Past, present, and future, 2021. doi:10.1109/JPROC.2021.3072740.

[3] Y. Chen, J. Yang, X. Dai, Q. Luo, F. Niu, Distributed model predictive formation control of micro multirotor uavs with virtual tube, IEEE Transactions on Aerospace and Electronic Systems 61 (2025) 9476–9489. doi:10.1109/TAES.2025.3549005.

[4] E. Soria, F. Schiano, D. Floreano, Predictive control of aerial swarms in cluttered environments, Nature Machine Intelligence 3 (2021) 545–554. URL: https://doi.org/10.1038/s42256-021-00341-y. doi:10.1038/s42256-021-00341-y.

[5] E. Soria, F. Schiano, D. Floreano, Distributed predictive drone swarms in cluttered environments, IEEE Robotics and Automation Letters 7 (2022) 73–80. doi:10.1109/LRA.2021.3118091.

[6] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, F. Gao, Swarm of micro flying robots in the wild, Science Robotics 7 (2022) eabm5954. URL: https://www.science.org/doi/abs/10.1126/scirobotics.abm5954. doi:10.1126/scirobotics.abm5954.

[7] Y. Bu, Y. Yan, Y. Yang, Advancement challenges in uav swarm formation control: A comprehensive review, Drones 8 (2024). URL: https://www.mdpi.com/2504-446X/8/7/320. doi:10.3390/drones8070320.

[8] V. Malke, O. Kosak, S. Rossi, M. Schörner, C. Wanninger, W. Reif, Swarmcage: Development sandbox for interactive swarms, formation flights, and continuous observation using crazyflies, ros, and vicon, Currently under review. (2026).

[9] J. Desai, J. Ostrowski, V. Kumar, Modeling and control of formations of nonholonomic mobile robots, IEEE Transactions on Robotics and Automation 17 (2001) 905–908. doi:10.1109/70.976023.

[10] V. Lesch, M. Breitbach, M. Segata, C. Becker, S. Kounev, C. Krupitzer, An overview on approaches for coordination of platoons, IEEE Transactions on Intelligent Transportation Systems 23 (2022) 10049–10065. doi:10.1109/TITS.2021.3115908.

[11] R. M. de Mendonça, N. Nedjah, L. de Macedo Mourelle, Swarm robots with queue organization using infrared communication, in: B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan (Eds.), Computational Science and Its Applications – ICCSA 2012, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012, pp. 136–147.

[12] J. Huang, Z. Ji, S. Xiao, C. Jia, Y. Jia, X. Wang, Multi-agent vehicle formation control based on mpc and particle swarm optimization algorithm, in: 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), volume 6, 2022, pp. 288–292. doi:10.1109/ITOEC53115.2022.9734371.

[13] Y. Yang, X. Xiong, Y. Yan, Uav formation trajectory planning algorithms: A review, Drones 7 (2023). URL: https://www.mdpi.com/2504-446X/7/1/62. doi:10.3390/drones7010062.

[14] L. Pichierri, A. Testa, G. Notarstefano, Crazychoir: Flying swarms of crazyflie quadrotors in ros 2, IEEE Robotics and Automation Letters 8 (2023) 4713–4720.

[15] O. Kosak, P. Kastenmüller, C. Wanninger, W. Reif, An approach for extended swarm formation flight with drones: Protease 2.0, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Rigorous Engineering of Collective Adaptive Systems, Springer Nature Switzerland, Cham, 2025, pp. 263–280.

[16] O. Kosak, P. Kastenmüller, W. Reif, Let's do the swarm flight again: unleashing the potential of protease 2.0 for drone formation flight, International Journal on Software Tools for Technology Transfer (2026). URL: https://doi.org/10.1007/s10009-025-00834-w. doi:10.1007/s10009-025-00834-w.

[17] O. Kosak, V. Malke, F. Nebel, P. Kastenmüller, W. Reif, Protease 2.0 with ros2: Parametrizable swarm formation flight using crazyflies, Currently under review. (2026).