

## An approach for extended swarm formation flight with drones: PROTEASE 2.0

Oliver Kosak, Philipp Kastenmüller, Constantin Wanninger, Wolfgang Reif

### Angaben zur Veröffentlichung / Publication details:

Kosak, Oliver, Philipp Kastenmüller, Constantin Wanninger, and Wolfgang Reif. 2025. "An approach for extended swarm formation flight with drones: PROTEASE 2.0." In Leveraging applications of formal methods, verification and validation: rigorous engineering of collective adaptive systems: 12th International Symposium, ISoLA 2024, Crete, Greece, October 27–31, 2024, proceedings, part II, edited by Tiziana Margaria and Bernhard Steffen, 263–80. Berlin: Springer. [https://doi.org/10.1007/978-3-031-75107-3\\_16](https://doi.org/10.1007/978-3-031-75107-3_16).

### Nutzungsbedingungen / Terms of use:

licgercopyright

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:

**Deutsches Urheberrecht**

Weitere Informationen finden Sie unter: / For more information see:

<https://www.uni-augsburg.de/de/organisation/bibliothek/publizieren-zitieren-archivieren/publiz/>



# An Approach for Extended Swarm Formation Flight with Drones: PROTEASE<sup>2.0</sup> <sup>★</sup>

Oliver Kosak<sup>1</sup>[0000-0003-0563-9797], Philipp Kastenmüller<sup>1</sup>[0009-0007-1342-5437],  
Constantin Wanninger<sup>1</sup>[0000-0001-8982-4740], and Wolfgang Reif<sup>1</sup>

Institute for Software & Systems Engineering (ISSE) [kosak@isse.de](mailto:kosak@isse.de)

**Abstract.** Drone formation flights, as performed in the Intel Drone Shows, demonstrate the current state of technology fascinatingly. We revisit this idea using the paradigm of self-organization in the form of swarm behavior. Applying swarm behavior for formation flight promises high scalability, robustness, and flexibility. Swarm behavior allows for impressive patterns where centrally coordinated approaches might reach their limits. In this paper, we propose PROTEASE<sup>2.0</sup> as an approach for parametrizable swarm behavior of the next level. Like its predecessor, PROTEASE<sup>2.0</sup> enables us to use a single generalized implementation for producing emergent effects by only adjusting parameters for the swarm members. Further, we now facilitate novel formations previously unattainable. New formations include parallel swarms interacting with each other, single swarms using multiple reference points enabling surprising flight patterns, and hierarchical swarm structures extending the possibilities even further. Our focus in this paper lies in the experimental evaluation of these concepts in simulated environments. In combination with successful pre-evaluations concerning swarm behavior using real drones, we confidently look towards future experiments also applying PROTEASE<sup>2.0</sup> in the real world.

**Keywords:** Swarms · Drones · Formation Flight · Multi-Robot Systems.

## 1 Introduction

In recent years, formation flight as performed by the Intel Drone Shows [10], has attracted an audience with its fascinating spectacle. These shows entail the coordinated motion of numerous drones, executing sophisticated maneuvers as they traverse the nighttime airspace. Through precise synchronization, these drones create formations that combine artistic creativity with state-of-the-art technological advancements. Orchestrated motions surpass conventional displays, combining innovation and aesthetics.

In this paper, we propose an approach aiming for the same goal but realizing it using the paradigm of self-organization, i.e., through swarm behavior applied

---

<sup>★</sup> Partially funded by DFG (German Research Foundation), grant number 402956354.

for drone formation flight. Leveraging swarm algorithms for formation flight represents a paradigm shift in the control of autonomous aerial systems, promising scalability, robustness, and flexibility. Mimicking nature seen in flocks of birds or schools of fish, swarm algorithms can empower autonomous vehicles to collaborate seamlessly in the skies [31]. The swarm approach achieves coordination without reliance on a central controller, ensuring decentralized decision-making for real-time adaptability in dynamic environments. The inherent fault tolerance of swarm algorithms enhances reliability, crucial for applications demanding operational resilience. Furthermore, the scalability of these algorithms allows for the effortless integration of additional members into formations which might be beneficial also for other industries where spatial distribution is of relevance, e.g., distributed surveillance [16, 17] or agriculture [5].

Unfortunately, there is an ongoing trend where the development of each new swarm application entails the creation of a unique software approach [7]. While these specialized approaches demonstrate effectiveness in their designated tasks, such as utilizing collective swarm behavior for search operations [34] or distributed surveillance [16, 18], users often face difficulties in adapting them to slightly different scenarios and hinder themselves from prior developments.

To address this challenge, we suggest adopting a standardized pattern that can encapsulate the swarm behavior of a specific class more broadly. Developers working with aerial multi-robot systems can integrate this pattern during the design phase and subsequently customize its parameters at runtime to attain distinct emergent effects. We previously identified such a common pattern in [14] and call it **PROTEASE** (Algorithmic Pattern for Trajectory-Based Swarm Behavior).<sup>1</sup> In this paper, we introduce **PROTEASE**<sup>2.0</sup> allowing for formation flight of the next level, e.g., multiple hierarchical layers of swarm behavior within a single swarm, interacting swarms, and improved single swarm behavior. That way, complex formation patterns can be defined within the set of parameters. While we focus on the visual impressions of the respective swarm behaviors here, we are convinced that our approach might also enhance the goal-oriented usage of swarm behavior in the next step, e.g., when applying varieties of particle swarm optimization (PSO) applied for search and rescue missions, as we proposed in [12]. Here, we focus on the required concepts and the algorithm design in an abstract simulation environment. We consider the integration of our findings with real-world drone formation flight as the consequent next step. Therefore, we already have an appropriate approach at hand [14] that integrates the multi-agent framework **Jadex** [4] used for high-level decisions with the robotics operating systems framework **ROS** [23] including a **Gazebo** physics simulation [11].

To give some background, we first briefly introduce the concept of **PROTEASE** in Section 2. We then give an insight into related work concerning parametrizable swarm behavior in Section 3. In Section 4, we then introduce our new approach of multi-layered, parametrizable swarm behavior we call **PROTEASE**<sup>2.0</sup>. We evaluate

---

<sup>1</sup> "... proteases are key regulators of a striking variety of biological processes ... they regulate different processes in response to developmental and environmental cues" [9].

our approach by example in a proof of concepts in Section 5 which we support by referencing our source code and video materials provided on GitHub. In Section 6, we then conclude on our results and give an outlook concerning possible next steps.

## 2 A Brief Description of Protease

PROTEASE is our approach for generalizing swarm behavior. The idea behind it is, on the one hand, to implement a general rule set for the behavior each individual in a swarm follows, that, on the other hand, can be parameterized externally to realize different swarm behavior. In doing so, our objective is to standardize various functionalities commonly employed by researchers to implement diverse swarm behaviors in aerial robotic systems. Despite yielding distinct emergent effects, it is apparent that swarm algorithms such as particle swarm optimization [34], the well-known flocking behavior analyzed in [24], shaping and formation algorithms [26], and distribution algorithms [16, 18] rely on a common set of local actions. These actions include measuring specific parameters, communicating with neighboring swarm members, and adjusting the flying robot's movement vector. For instance, to achieve flocking behavior as described in [24], each "boid" must execute a specific combination of functions, including position and velocity measurements, information exchange with other swarm members, and subsequent adjustment of its movement vector, resulting in the collective emergence of flock-like behavior among individual agents.

Based on this discovery, we established the parameters for PROTEASE (cf. Fig. 1) essential for generating a broad spectrum of swarm behaviors [14]. Thereby, we had in mind to enable the integration of PROTEASE into approaches for task orchestration, e.g., such that we introduced in [13]. According to this, the parameters forming the behavior of PROTEASE are these four swarm functions:

- **A** (aggregation), enabling the swarm to determine the collective result.
- **T** (termination), letting the swarm determine that it has reached its goal.
- **G** (grouping), defining each agent's neighborhood for information exchange.
- **C** (calculation), determining how the information should be processed and mapped to a goal trajectory for further movement.

Each participating agent then cyclicly executes the same activity as depicted in Fig. 1, i.e., by 1st) measuring all relevant data necessary for **A** and **C**, 2nd) exchanging that measures with all neighbors encoded in **G**, 3rd) calculating a new trajectory using **C**, 4th) adapting the trajectory, and 5th) terminating the rules execution if the termination criteria encoded in **T** holds or starting over with 1st) otherwise. We call each of this cycles an execution round.

Examples of swarm behavior that can be realized by executing PROTEASE with varying parameters, denoted as ATGC<sup>2</sup>, are, e.g., Boiding according to [24],

<sup>2</sup> In the case of DNA, "each nucleic acid contains four ... nitrogen-containing bases: adenine (A), guanine (G), cytosine (C), [and] thymine (T)" [25]. "Nucleic acids are the main information-carrying molecules ... they determine the inherited characteristics of every living thing" [25].

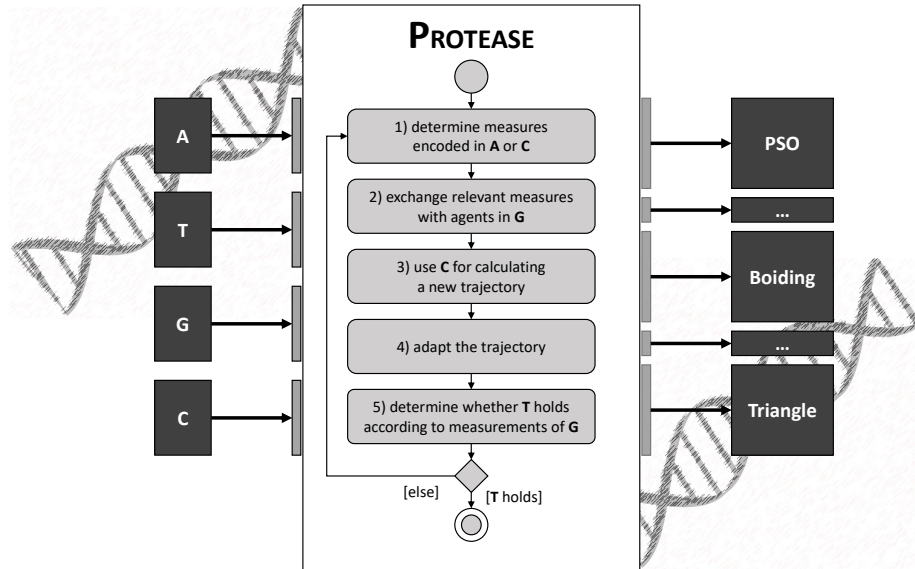


Fig. 1: PROTEASE is defined as a local behavior, each agent in a swarm executes.

Triangle-Formation according to [16], or PSO-like [34] search for the spot of highest concentration of a continuously distributed parameter  $P$  (with  $A$  evaluating the centroid regarding the positions of all agents in the swarm,  $T$  evaluating if the geometric diameter regarding the positions of agents is below a threshold,  $G$  returning the relevant agents in the swarm, and  $C$  evaluating the typical local rules of PSO involving the specific parameter  $P$ ). We can further express other swarm behavior with parameters ATGC and execute them with PROTEASE, which we investigated in [14].

To align with our approach PROTEASE, swarm behavior must adhere to certain key principles. Therefore, we must consider the following assumptions. The swarm behavior must function independently of environmental markings, differentiating it from ant-colony-optimization-based swarm algorithms using stigmergy [3]. This requirement stems from our original intention for PROTEASE to operate within aerial swarms comprising real drones [32], making the implementation of stigmergy challenging. Achieving stigmergy would necessitate a digital representation of the environment accessible to all agents executing PROTEASE simultaneously. While we are exploring the feasibility of incorporating such functionality, we aim to abstract from stigmergic swarm behavior in the current context. Our emphasis is particularly on emergent effects stemming from swarm behavior that results from adjustments to the trajectories of participating agents. To ensure the practical implementation of PROTEASE within real-world flying swarms, it is imperative that each agent can potentially communicate with any other agent in the swarm. This capability is essential for enabling self-termination mechanisms in certain types of swarm behavior (e.g.,

the termination of a PSO execution), while it does not affect the actual execution of local swarm rules which can still be based on the local neighborhood. Further, in the real world, we cannot assume to have local sensors available for all relevant spatial values, e.g., the precise measurement of other agents' positions. Last, we require all agents executing PROTEASE to be able to move in 3-dimensional space.

### 3 Related Work

The literature exploring swarm behavior, swarm algorithms, and swarm intelligence is extensive and diverse. When considering the application of swarm behavior in real-world scenarios, researchers typically pursue two main groups of approaches.

*Specific Swarm Behavior* The one group of approaches focuses on a specific swarm behavior each, which is then analyzed and adapted for use in technical systems. Numerous examples of this approach exist, although only a selection of relevant research can be discussed here. For solving different instances of the search problem, there exists a multitude of applications of PSO applied to different use cases [34]. Some of them even include flying robots used in the real world [19, 28] and simulated environments [15, 27]. Other examples show that we can also adapt the self-coordination abilities of swarms of birds or fish to technical systems for reducing the coordination efforts which can be a useful property when commanding huge groups of aerial robots [19, 28, 31]. There also exist approaches that aim at generating software controllers for individual swarm agents with genetic algorithms and other learning techniques, e.g., for coordinated motion in general [29], foraging tasks [20], or searching and acting tasks [8]. To achieve a nearly uniform distribution of swarm entities within a specified area, such as in the context of Distributed Surveillance, Ma et al. [18] have tailored a deployment algorithm based on potential fields. However, this algorithm is limited in its applicability to that specific use case. Meanwhile, Li et al. [16] propose the adaptation of their swarm approach for Distributed Surveillance to also encompass flocking and obstacle avoidance, yet they do not delve further into exploring this direction. We view this as a promising step towards establishing a generalized pattern for realizing swarm behavior, which aligns with our approach. In a study by Garcia et al. [28], the authors modify the PSO algorithm for the deployment of flying robots in disaster scenarios to conduct area exploration and detect victims. Despite the ability to adjust parameters to suit different objectives, this approach remains confined to the narrowly defined scope of its application and is not easily extendable. Similarly, in a work by Vasarhelyi et al. [31], an adapted flocking algorithm inspired by the principles outlined in [24] is demonstrated to enable flying robots to exhibit swarm behavior closely resembling that of natural swarms. However, the implementation of this algorithm is highly specific and can only achieve the predetermined swarm behavior. In their work, Dedousis et al. [7] introduce swarm primitives tailored for managing

multi-robot systems through their approach PaROS. Although their objective is to implement swarm behavior for particular tasks using these primitives, their focus does not extend to generalizing swarm behavior comprehensively.

*Generalizing Swarm Behavior* Another group of approaches focuses on developing generalized frameworks for collective behavior rather than targeting specific applications. These frameworks can be adapted to various scenarios and requirements. Protelis [21] and its successor MacroSwarm [1] are examples of this approach. They conceptualize entities within a collective system as elements within vector fields, enabling collective programming by manipulating operations within these fields. Changes are disseminated through implicit communication among entities, allowing complex collective behaviors to be implemented abstractly. However, Protelis faces challenges in prototyping 3D swarm behavior due to limited integration with simulation frameworks and drones, and it currently lacks support for all features of PROTEASE<sup>2.0</sup>, such as hierarchical swarm behavior. Buzz [22] is another framework designed for generalizing and programming collective behavior. In Buzz, swarms are treated as primary abstractions, with tasks allocated among individual robots based on local interactions. Buzz uses virtual stigmergy with distributed tuple spaces to achieve swarm-wide consensus and allows for extensibility through new primitives. However, it does not support hierarchical swarm behavior as proposed with PROTEASE<sup>2.0</sup>. Meld, developed by Rollman et al. [2], also aims to manage swarms through high-level abstractions. Yet, Meld’s execution model is not well-suited for real-world flying swarms, with computations for ensemble behaviors taking significant time and the robots’ limited capabilities making practical implementation challenging. Thus, Meld’s execution model is more restricted compared to our approach. Lastly, Varughese et al. [30] propose a design paradigm focused on generalizing swarm behavior by minimizing communication requirements. Their approach, which emphasizes minimalistic and decentralized functionality, suggests that combinations of primitives can enable complex behaviors like collective transport in 2D environments. However, their paradigm relies on cyclic synchronization messages, leading to lengthy stabilization times (e.g., 1400 seconds for distributed localization), which may impact its effectiveness for flying swarms.

## 4 Approach

The PROTEASE<sup>2.0</sup> approach extends PROTEASE by utilizing the concepts of multiple reference points and swarm agents that can act as such reference points. We use the latter to create a multi-layered, parametrizable swarm behavior. Thereby, we also distinguish between the use of a single set of agents and the use of multiple sets of agents. As we focus on the visual impression of swarm behavior in this paper, we choose the parameters for PROTEASE<sup>2.0</sup> respectively: For the extensions we describe within this paper, we currently neglect the auto-termination possibility of PROTEASE<sup>2.0</sup> we typically use for mission integration [13, 14]. Thus, the aggregation function  $A$  and the termination function  $T$  are not considered. For

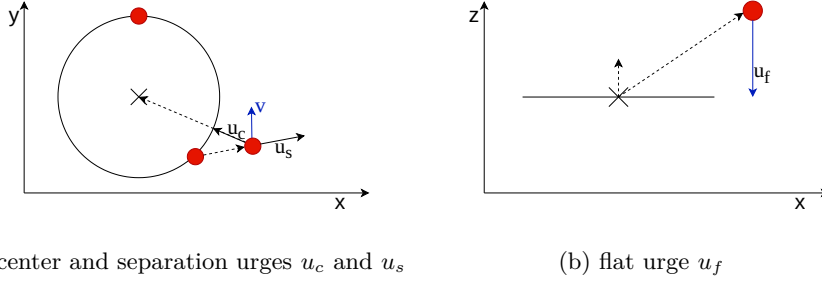


Fig. 2: Schematic for calculating urges in  $\mathcal{C}_{ring}$ . Dashed arrows represent measured values for one agent in the red swarm. Solid arrows show calculated urges  $u_c$ ,  $u_s$ , and  $u_f$ . The blue arrow indicates the resulting trajectory  $v$ .

the grouping function  $\mathbf{G}$  we select all neighbors within a predefined radius around the given agent. The calculator function  $\mathbf{C}$  stays freely configurable in general, but for the sake of clarity, we restrict our descriptions to one single calculation function producing a ring swarm behavior we call Ring-of-Fliers.

#### 4.1 Running example: The Ring-of-Fliers

To establish a ring formation we use as a running example throughout the rest of our descriptions, we define a calculation function  $\mathcal{C}_{ring}$  producing the desired swarm behavior. We provide each agent participating in the swarm with the required information as an input for  $\mathcal{C}_{ring}$ . This includes knowledge 1) about its position, 2) the 6-dimensional transformation of a user-controlled reference point that denotes the center of the desired ring formation, and 3) a ring-specific parameter  $r$  that is utilized to define the intended radius of the ring. The reference point can be realized by a virtual element in simulation or as an actual physical device manipulated in the real world (e.g., an additional user-controlled drone). Other relevant information for executing  $\mathcal{C}_{ring}$  concerning the agent's neighbors arrive automatically as a result of the multi-cast-like communication happening among all neighbors in  $\mathbf{G}$  (cf. Fig. 1). The ring formation then establishes through the local combination of three different urges  $u_c$ ,  $u_s$ , and  $u_f$  for each agent (cf. Eqs. (1a) to (1c)), following the swarm model of [24] that combines different *urges* addressing specific local rules (e.g., cohesion urge, separation urge, and alignment urge) to a new local decision for the agent's next trajectory in 3-D. In the calculation for the Ring-of-Fliers pattern described here, each urge is represented by a vector indicating the desired trajectory for an agent: The center urge  $u_c$  points towards the measured position describing the center of the ring, i.e., the position of the reference point. The separation urge  $u_s$  points away from the closest known neighbor covered by the grouping function  $\mathbf{G}$ . The flat urge  $u_f$  points towards the closest position on the plane given by the reference point and its orientation as a normal vector. Expressions  $c_{own}$  and  $c_{ref}$  describe the agents' capabilities for measuring the 6-dimensional transformation of itself

and the reference point, respectively and both return a 6-D vector. In Eqs. (1a) to (1c) and (2c), the methods  $\text{POS}()$  and  $\text{ROT}()$  provide the position and rotation as a unit vector from a 6-dimensional transformation, respectively. Furthermore,  $\text{CLOSE}()$  provides the transformation to the closest Agent in  $\mathbf{G}$  concerning the Euclidean distance. The symbol  $\circ$  represents the scalar product of vectors. The function  $\mathbf{C}_{ring}$  then calculates the resulting trajectory  $v$  using the weighted sum of the urges with the respective weights  $\omega_c$ ,  $\omega_s$ , and  $\omega_f$  for a new agent movement vector  $v := \omega_c \cdot u_c + \omega_s \cdot u_s + \omega_f \cdot u_f$ :

$$u_c := (\text{POS}(c_{ref}) - \text{POS}(c_{own})) \quad (1a) \qquad \omega_c := \frac{|u_c| - r}{|u_c|} \quad (2a)$$

$$u_s := \text{POS}(c_{own}) - \text{POS}(\text{CLOSE}(\mathbf{G})) \quad (1b) \qquad \omega_s := \frac{1}{|u_s|^2} \quad (2b)$$

$$u_f := \text{ROT}(c_{ref}) \quad (1c) \qquad \omega_f := -((\text{POS}(c_{own}) - \text{POS}(c_{ref})) \circ \text{ROT}(c_{ref})) \quad (2c)$$

## 4.2 Extension with Multiple Reference Points

While we describe the cyclic 'default' calculation model for  $\text{PROTEASE}^{2.0}$  (according to our previous findings in [12] and [14]) in Section 4.1, within this section we extend the concepts of  $\text{PROTEASE}$  for using multiple reference points at the same time. By doing so, we can achieve novel ring-based formation flight patterns in two different ways which we schematically describe in Fig. 3. All agents still perform their calculations in execution rounds that only terminate, if an independent termination signal is received.

*1) Single Set Approach:* By extending the results of  $c_{ref}$  (that measures the 6-D transformation of the reference point) from one measured value  $ref$  to a set of measurements  $\text{REF}$ , each agent now has to decide for one of these values when calculating  $u_c$ , and  $u_f$ . While also other selections are possible, we choose to use the closest one concerning the respective agent's current position, extending  $\mathbf{C}_{ring}$  to  $\mathbf{C}_{ring*}$  (cf. Algorithm 1). Because the selection in Algorithm 1 is renewed within each execution round of  $\text{PROTEASE}^{2.0}$  (cf. the execution model in Section 2), each agent consequently possibly updates its reference point in each round. For achieving that, we now use  $\text{CLOSE}(\text{REF})$  in  $\mathbf{C}_{ring*}$  to return the transformation of the (euclidean distance) closest reference point for each entry  $c_{ref}$  in Eqs. (1a), (1c) and (2c). Thus, the respective reference point each agent considers as relevant may change throughout the execution time of  $\text{PROTEASE}^{2.0}$ , e.g., when reference points adapt their position over time based on user or environmental interactions. To achieve the desired effect and to reduce the danger of collisions within the swarm, we leave the separation urge  $u_s$  unchanged to points away from the closest neighbor within  $\mathbf{G}$ . In Fig. 3b we depict the new calculation for  $\mathbf{C}_{ring*}$  in case of two different reference points.

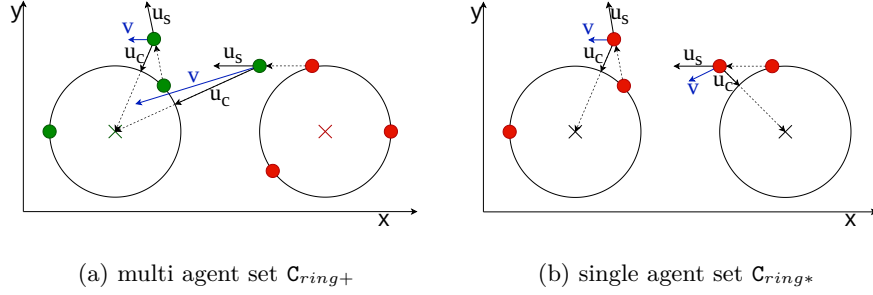


Fig. 3: Extensions to the Ring-of-Fliers behavior that involve multiple reference points, showing the center urge  $u_c$  and the separation urge  $u_s$  in black for two agents each to illustrate the difference between with  $C_{ring+}$  and  $C_{ring*}$ . In Fig. 3a, green agents are assigned to the green reference point, while red agents are assigned to the red one. In Fig. 3b, all agents belong to the same agent set.

---

**Algorithm 1** reference point selection in  $C_{ring*}$ , executed each round.

---

- 1: **for all** agents  $a$  in  $G$  **do**
  - 2:      $a.c_{ref} = \text{CLOSE}(\text{REF})$
  - 3: **end for**
- 

2) *Multi Set Approach*: In a second adaptation of  $C_{ring}$  involving multiple possible reference points, i.e.,  $C_{ring+}$ , we explicitly assign each agent to one of the available reference points, i.e., restrict its local view to only that reference point **AGENT-REF**. This also defines the number of possible agents in each ring throughout the overall execution of PROTEASE<sup>2.0</sup>. Instead of letting each agent choose one of the reference points in **REF**, we use that specific reference point when calculating  $u_c$ , and  $u_f$  in all execution rounds of PROTEASE<sup>2.0</sup> (cf. Algorithm 2). Thus, also when reference points change their position over time each agent considers only its assigned reference point even when other reference points are positioned closer. Similar to  $C_{ring*}$  in, we leave the separation urge  $u_s$  unchanged. In Fig. 3a, we depict the new calculation for  $C_{ring+}$  in case there are two different reference points but agents are explicitly assigned to reference points.

---

**Algorithm 2** reference point selection for  $C_{ring+}$ , executed once at start

---

- 1: **AGENT-REF** := oneof **REF**
  - 2: **for all** agents  $a$  in  $G$  **do**
  - 3:      $a.c_{ref} = \text{CLOSE}(\text{AGENT-REF})$
  - 4: **end for**
-

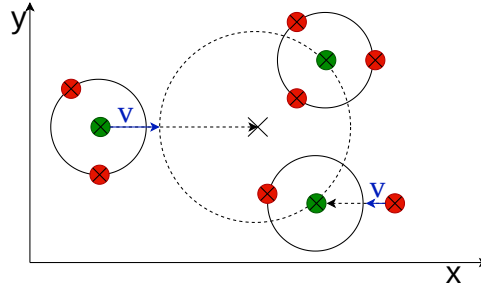


Fig. 4: Hierarchic concept of PROTEASE<sup>2.0</sup> using the  $C_{ringH}$  parameter, structured as a 2-layered Ring-of-Rings. Agents with crosses can serve as reference points in  $C_{ring*}$ . Only the resulting trajectory  $v$  is shown. Colors depict the different layers of the swarm.

### 4.3 Extension with Hierarchies

In addition to the extension with multiple reference points, we further introduce the concept of hierarchies within swarms with an adapted calculation function  $C_{ringH}$ . Compared to  $C_{ring}$  and  $C_{ring*}$ , where reference points are user-controlled, we extend the set of reference points to also include all swarm agents themselves in  $C_{ringH}$ . Thus, when assigning agents to reference points, it is also possible to assign them to other agents within the swarm. That way, we can create hierarchies of sub-swarms by appropriately allocating agents to reference points. In this allocation, we need to avoid reflexive allocations and only wisely decide on cyclic allocations, as the effects might end up in chaotic swarm behavior. On each layer of the resulting hierarchy, we now have a set of agents collectively producing the desired swarm behavior. For the top-level layer 0, we use a user-controlled reference point, allowing us to control the whole system. For all higher-level layers  $i \in \{1..n\}$  we then allocate an adequate number of agents to use one of the agents on layer  $i - 1$  as their reference point. Thus, each agent  $\alpha_x$  on layer  $i$  with  $i > 0$  uses an agent  $\alpha_y \in \mathcal{A}_{i-1}$  as its reference point, where  $\mathcal{A}_i$  is the set of agents on layer  $i$ .

We then can assign reference points to agents in a fixed way similar to  $C_{ring+}$ , creating stable hierarchical structures. As an alternative and similar to  $C_{ring*}$  in Section 4.2, we can also leave the decision for a specific reference point to the agent. Then, we only need to assign agents to layers in the hierarchy, leaving more control to the system while reducing the possibilities for controlling the visual effects on the collective level.

As an additional feature of hierarchic swarms, reference points also can provide an individual calculation function, further extending the possible collective effect. Thus, we can mix different formations where participating agents might also change their behavior according to their current reference point (that might change over time depending on the respective calculation function provided by

the reference point). For the sake of simplicity, our description in Fig. 4 depicting a two-layered hierarchic execution of PROTEASE<sup>2.0</sup> involves our running example  $C_{ring}$  only. Nevertheless, we cover this feature within our reference implementation we use for experiments provided in Section 5.

## 5 Proof of Concepts

We already validated the generality of PROTEASE in its original version in [14]. There, we analyzed PROTEASE by employing comparable parameters  $A$ ,  $T$ ,  $G$ , and  $C$  to achieve diverse emergent effects, inducing corresponding modifications in swarm behavior. In the evaluation in this paper, we now analyze the extended concepts of PROTEASE<sup>2.0</sup> concerning multiple reference points and its possibility to hierarchically layer the swarm behavior. To conduct these assessments, we utilize the NetLogo multi-agent programmable modeling environment<sup>3</sup>.

### 5.1 Evaluation Environment

NetLogo [6] is a programming environment tailored for facilitating the setup of Multi-Agent system simulations. It abstracts away many intricate details, simplifies complexity through appropriate discretization, and reduces the barrier to prototyping collective and swarm behavior. NetLogo allows for simulating agents, 3-dimensional environments, and logical time for state progression.

Within this environment, agents are situated and oriented. They possess an awareness of environmental data and can utilize it for internal processing. Furthermore, agents can communicate with each other and exchange any information they possess in an unlimited fashion. NetLogo uses a cyclic execution engine that allows agents to execute calculations in a round-based fashion. In our illustrations, agents are depicted by arrows, denoting their position and orientation in the 3-dimensional environment. The color of the arrows provides additional information, such as the agent set or the layer the agents belong to.

For evaluating our concepts from Section 4, we use two different NetLogo models which we also provide on GitHub<sup>4</sup>. We designed one model *MultiInput* for the assessment of an extension with multiple reference points, while we intend the other (*Hierarchic*) for the evaluation of layered swarm execution. The user interface for *MultiInput* enables the user to define the initial number of agents and reference points. While our concepts are not limited to a number, the implementation supports a limited number (five) of reference points only. These reference points can be moved individually or by using the presets. Additionally, the implementation allows for the modification of agent-specific properties, such as defining their maximum velocity ( $velocity_{max}$ ) and communication radius ( $com-radius$ ). The user can modify the specific swarm behavior by adjusting the provided parameters, such as the radius of the Ring-of-Flyers.

<sup>3</sup> NetLogo download at <https://ccl.northwestern.edu/netlogo/download.shtml>

<sup>4</sup> PROTEASE<sup>2.0</sup> on GitHub at <https://github.com/OliverISSE/Protease2.0>

Finally, the user can choose between using a single agent set or if the agents are evenly distributed among the reference points initially. *Hierarchic* provides similar features. In addition, it allows for specifying different swarm behavior for each hierarchic layer, including the swarm type, the number of agents on the hierarchy layer, and the communication radius. Again and for the sake of clarity, while our concepts are not limited to a specific number, the implemented model supports a limited number of hierarchic layers only.

## 5.2 Experiments

We provide multiple example configurations for our provided NetLogo implementation of PROTEASE<sup>2.0</sup>. While these already visualize interesting complex swarm behavior, we advise the readers to execute their experiments using our code provided on GitHub to identify further emergent effects arising from the combination of different swarm behavior. Our experiments within this paper do not explicitly investigate the typical swarm properties *robustness* and *scalability*. Nevertheless, these properties also hold for PROTEASE<sup>2.0</sup> unless their excessive usage reduces the visual effects we put on emphasis in this paper. To convince the readers of the general functionality, we extended our implementation provided on GitHub with the respective functionality. While the figures depicted here only provide a 2-dimensional view of the respective swarm behavior, all experiments were performed in the provided 3-dimensional environment. Because we detected significant qualitative differences in the produced swarm effects, we investigate these differences within our experiments by comparing the emergent effects of the different calculation functions  $C_{ring+}$  (Single Set Approach) and  $C_{ring*}$  (Multi Set Approach), as we describe them in Section 4.2.

### 5.3 Multiple Reference Points: The Flower Pattern

In the first experiment, we first utilize the single agent set approach  $C_{ring*}$ . We align five reference points in a cross formation. For each reference point, we decide on the Ring-of-Fliers swarm behavior. Due to the agents' behavior of always choosing the closest reference point, the rings don't close. The result is a flower-like shape, where agents evenly distribute on the edge line of the shape (cf. Fig. 5a). Switching to the multiple set approach with  $C_{ring+}$  while leaving all other parameters untouched, forces the agents to create one ring for each reference point. While all rings then close also the inner part of the shape, we can observe their interference obfuscating the clear shape (cf. Fig. 5b). We provide the comprehensive set of parameters in Table 1 allowing for the reproduction of the experiment using our implementation provided on GitHub.

### 5.4 Multiple Reference Points: The Olympic Ring Pattern

In a second experiment, we again first utilize the single agent set approach  $C_{ring*}$ . By aligning five reference points in a W-like formation where all use the Ring-of-Fliers swarm behavior, we can observe the swarm creating a pattern similar to

Table 1: Comprehensive set of parameters for the first (Flower Pattern) and the second (Olympic Pattern) experiment.

experiment	# agent	com-radius	velocity <sub>max</sub>	function	r <sub>c<sub>ring</sub></sub>
flower-multi	45	16	0.2	$C_{ring*}$	6
flower-single	45	16	0.2	$C_{ring+}$	6
olympic-multi	50	16	0.2	$C_{ring*}$	5
olympic-single	50	16	0.2	$C_{ring+}$	5

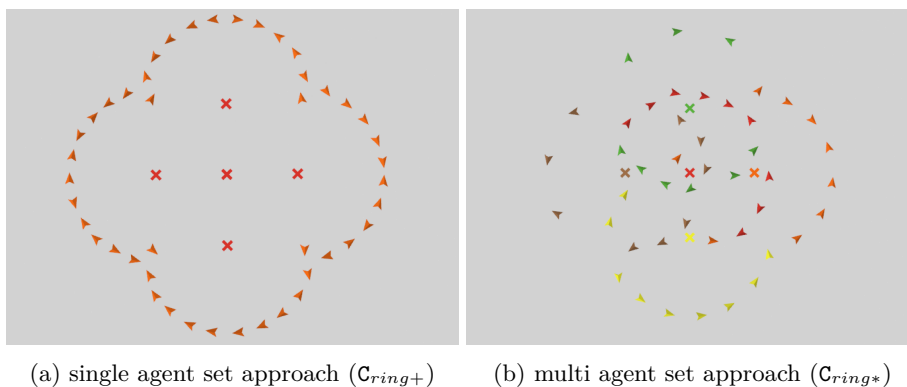


Fig. 5: Flower Pattern experiment producing a flower-like visual effect.

the famous Olympic Rings (cf. Fig. 6a). When switching to the multiple agent set approach with  $C_{ring+}$ , we can again identify a qualitative difference. When we do not force agents to permanently assign one reference point, agents do not distribute evenly on the different rings, reducing the visual effect (cf. Fig. 6b). We provide the comprehensive set of parameters in Table 1 allowing for the reproduction of the experiment using our implementation provided on GitHub.

### 5.5 Hierarchies: Ring-of-Rings

In a third experiment, we demonstrate the visual effects of the hierarchic swarm concept (cf. Section 4.3). For the sake of clarity, we restrict the amount of different hierarchic layers to three in this experiment. Nevertheless, our example implementation supports up to five hierarchic layers, coming closer to the in-general unrestricted amount of hierarchic layers. To illustrate the effects, we again use the Ring-of-Fliers swarm behavior on all hierarchic layers and for all reference points. Switching from our one single agent set approach to our multiple set approach again produces different effects. As within the second experiment (cf. Section 5.4) the use of the single set approach can result in unbalanced rings (cf. Fig. 7a) while the multiple set approach creates balanced rings (cf. Fig. 7b). We provide the comprehensive set of parameters in Table 2 allowing for the reproduction of the experiment using our implementation provided on GitHub.

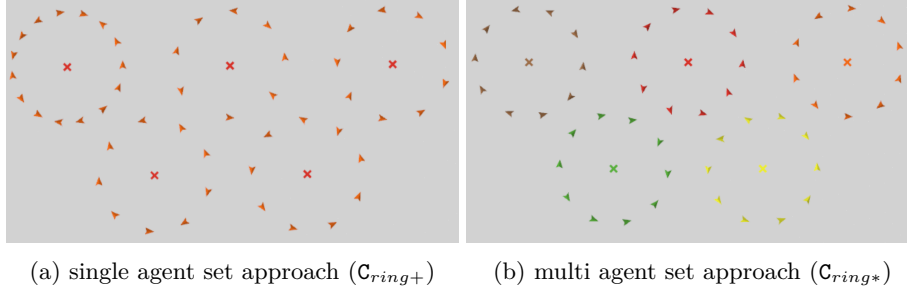


Fig. 6: Olympic Pattern experiment producing the effect of the Olympic Rings.

Table 2: Parameters of the third experiment. # agent indicates the total amount of agents and the count of agents per reference point in parentheses.

hierarchic layer	# agents	com-radius	velocity <sub>max</sub>	calc-function	$r_{C_{ring}}$
layer 0	4 (4)	25	0.2	$C_{ring+} \setminus C_{ring*}$	20
layer 1	16 (4)	20	0.2	$C_{ring+} \setminus C_{ring*}$	15
layer 2	128 (8)	10	0.2	$C_{ring+} \setminus C_{ring*}$	10

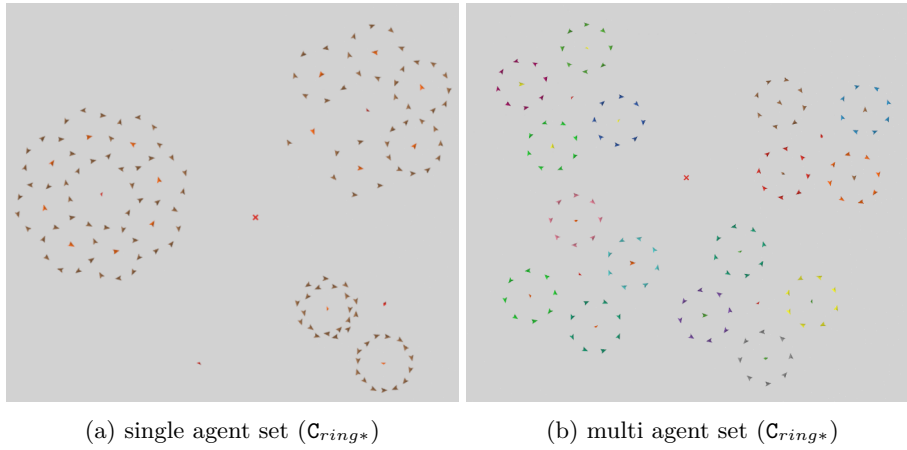


Fig. 7: Third experiment producing a solar system-like visual effect using the hierarchic swarm concept.

## 5.6 Discussion and Real-World Limitations

While we have already successfully implemented less sophisticated swarm behavior for real drones, i.e., single reference point, and flat layered swarms, we aim at also realizing the concepts presented in this paper with real drones. Bridging the reality gap will force us to restrict the swarm behavior and the resulting individual and collective movements in some way avoiding physical issues typically hindering unrestricted execution in the real world. While trajectories in simulated swarms do not cause downwash with their rotors, we need to take this into account when deploying respective behavior for real drones. While we can take care of possible issues using adequate collision hulls in our collision avoidance behavior we additionally use when executing real-world experiments, this might limit the visual effects we can see in our simulated experiments (cf. Section 5.2). To exactly reproduce the behavior, we would require excessive space enabling 'one-over-the-other' flight constellations. For smaller settings, e.g., in our flight arena [33], we might be forced to restrict the degree of freedom we allow a user when controlling the position of reference points or adjusting the parameters of PROTEASE<sup>2.0</sup>.

## 6 Conclusion and Future Work

In this paper, we introduced new swarm concepts with PROTEASE<sup>2.0</sup> for extending the possible range of decentralized formation flight patterns of drones. First, we refined our notion of parametrizable swarm behavior, where different emergent effects can result from one generalized implementation by only changing the parameters each swarm member uses. Then, starting with the formalization of a modified swarm algorithm realizing a ring formation around a reference point in a 3-dimensional environment, we demonstrated how we can achieve new formations to be produced by swarms in three different ways with PROTEASE<sup>2.0</sup>: (1) by increasing the number of parallel swarms with  $\mathcal{C}_{ring+}$ , (2) by changing the single reference point into a set of those, and (3) by introducing hierarchic swarms. While we currently focus on the experimental evaluation of these new swarm concepts achieving novel flight patterns for drones in simulation, we already achieved proofs-of-concept using real drones forming simpler swarm patterns, e.g., the Ring-of-Fliers described in Section 4.1. In future work, we will tackle the physical challenges occurring when using real drones, e.g., battery constraints, downwash of rotors, and collision avoidance. While we already addressed these issues separately in previous research [14], we now will integrate them with PROTEASE<sup>2.0</sup>. Other future work can focus on improving the discovery abilities of our swarm behavior, e.g., concerning hierarchical Particle Swarm Optimization executions searching for certain parameters with each particle forming its own swarm aggregating and communicating local results to possibly better survey greater regions.

## Acknowledgment

The authors would like to thank all reviewers for their valuable suggestions.

## References

1. Aguzzi, G., Casadei, R., Viroli, M.: Macroswarm: A field-based compositional framework for swarm programming. In: Coordination Models and Languages: 25th IFIP WG 6.1 International Conference, COORDINATION 2023, Held as Part of the 18th International Federated Conference on Distributed Computing Techniques, DisCoTec 2023, Lisbon, Portugal, June 1923, 2023, Proceedings. p. 3151. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-3-031-35361-1\\_2](https://doi.org/10.1007/978-3-031-35361-1_2), [https://doi.org/10.1007/978-3-031-35361-1\\_2](https://doi.org/10.1007/978-3-031-35361-1_2)
2. Ashley-Rollman, M.P., Goldstein, S.C., Lee, P., Mowry, T.C., Pillai, P.: Meld: A declarative approach to programming ensembles. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2794–2800 (2007). <https://doi.org/10.1109/IR0S.2007.4399480>
3. Bianchi, L., Gambardella, L.M., Dorigo, M.: An ant colony optimization approach to the probabilistic traveling salesman problem. In: International Conference on Parallel Problem Solving from Nature. pp. 883–892. Springer (2002)
4. Braubach, L., Pokahr, A.: Developing distributed systems with active components and jadex. Scalable Computing: Practice and Experience pp. 100–120 (2012)
5. Carbone, C., Garibaldi, O., Kurt, Z.: Swarm robotics as a solution to crops inspection for precision agriculture. KnE Engineering **3**(2), 552–562 (Feb 2018). <https://doi.org/10.18502/keg.v3i1.1459>, <https://knepublishing.com/index.php/KnE-Engineering/article/view/1459>
6. CCL: Netlogo - northwestern’s center for connected learning and computer-based modeling (ccl). Available at <https://ccl.northwestern.edu/netlogo/>, accessed on 2020-08-28 (aug 2020)
7. Dedousis, D., Kalogeraki, V.: A framework for programming a swarm of uavs. In: Proceedings of the 11th PErvasive Technologies Related to Assistive Environments Conference. pp. 5–12 (2018)
8. Dorigo, M., Floreano, D., Gambardella, L.M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A.L., Decugniere, A., Caro, G.D., Ducatelle, F., Ferrante, E., Forster, A., Gonzales, J.M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., de Oca, M.M., O’Grady, R., Pinciroli, C., Pini, G., Retornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stutzle, T., Trianni, V., Tuci, E., Turgut, A.E., Vaussard, F.: Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. IEEE RAM **20**(4), 60–71 (2013). <https://doi.org/10.1109/MRA.2013.2252996>
9. Van der Hoorn, R.A.: Plant proteases: from phenotypes to molecular mechanisms. Annu. Rev. Plant Biol. **59**, 191–223 (2008)
10. Intel: Aerial Technology Light Show. Available at <https://www.intel.de/content/www/de/de/technology-innovation/aerial-technology-light-show.html>, accessed on 2021-02-01 (Feb 2021), <https://www.intel.de/content/www/de/de/technology-innovation/aerial-technology-light-show.html>
11. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent

- Robots and Systems (IROS) (IEEE Cat. No.04CH37566). vol. 3, pp. 2149–2154 vol.3 (2004). <https://doi.org/10.1109/IROS.2004.1389727>
12. Kosak, O., Bohn, F., Eing, L., Rall, D., Wanninger, C., Hoffmann, A., Reif, W.: Swarm and Collective Capabilities for Multipotent Robot Ensembles, currently under review. In: 9th Int. Symp. On Leveraging Appl. of Formal Methods, Verification and Validation (Oct 2020)
  13. Kosak, O., Huhn, L., Bohn, F., Wanninger, C., Hoffmann, A., Reif, W.: Maple-Swarm: Programming Collective Behavior for Ensembles by Extending HTN-Planning. In: 9th Int. Symp. On Leveraging Appl. of Formal Methods, Verification and Validation (Oct 2020)
  14. Kosak, O.: Mission programming for flying ensembles: combining planning with self-organization. doctoralthesis, Universität Augsburg (2021)
  15. Lee, K.B., Kim, Y.J., Hong, Y.D.: Real-time swarm search method for real-world quadcopter drones. *Applied Sciences* **8**(7), 1169 (2018)
  16. Li, X., Ercan, M.F., Fung, Y.F.: A triangular formation strategy for collective behaviors of robot swarm. In: Gervasi, O., Taniar, D., Murgante, B., Laganà, A., Mun, Y., Gavrilova, M.L. (eds.) *Computational Science and Its Applications – ICCSA 2009*. pp. 897–911. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
  17. Liu, Y., Liu, H., Tian, Y., Sun, C.: Reinforcement learning based two-level control framework of uav swarm for cooperative persistent surveillance in an unknown urban area. *Aerospace Science and Technology* **98**, 105671 (2020)
  18. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. *IEEE Transactions on Computers* **56**(7), 946–847 (2007)
  19. Na, H.J., Yoo, S.: Pso-based dynamic uav positioning algorithm for sensing information acquisition in wireless sensor networks. *IEEE Access* **7**, 77499–77513 (2019). <https://doi.org/10.1109/ACCESS.2019.2922203>
  20. Pérez, I.F., Boumaza, A., Charpillet, F.: Learning collaborative foraging in a swarm of robots using embodied evolution. In: *Artificial Life Conference Proceedings 14*. pp. 162–161. MIT Press (2017)
  21. Pianini, D., Viroli, M., Beal, J.: Protelis: Practical aggregate programming. In: *Proc. of the 30th Annual ACM Symposium on Applied Computing*. pp. 1846–1853. SAC '15, ACM (2015). <https://doi.org/10.1145/2695664.2695913>, <http://doi.acm.org/10.1145/2695664.2695913>
  22. Pinciroli, C., Beltrame, G.: Buzz: An extensible programming language for heterogeneous swarm robotics. In: *2016 IEEE/RSJ Int. Conf. on Intel. Robots and Systems (IROS)*. pp. 3794–3800 (Oct 2016). <https://doi.org/10.1109/IROS.2016.7759558>
  23. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: *ICRA workshop on open source software*. vol. 3, p. 5. Kobe (2009)
  24. Reynolds, C.W.: Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH computer graphics* **21**(4), 25–34 (1987)
  25. Roberts, R.J.: Nucleic acid. *Encyclopedia Britannica*. Available at <https://www.britannica.com/science/nucleic-acid>, accessed on 2020-06-14 (Feb 2020), <https://www.britannica.com/science/nucleic-acid>
  26. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. *Science* **345**(6198), 795–799 (2014)
  27. Skrzypecki, S., Tarapata, Z., Pierzchała, D.: Combined pso methods for uavs swarm modelling and simulation. In: Mazal, J., Fagiolini, A., Vasik, P. (eds.) *Modelling and Simulation for Autonomous Systems*. pp. 11–25. Springer International Publishing, Cham (2020)

28. Sánchez-García, J., Reina, D., Toral, S.: A distributed pso-based exploration algorithm for a uav network assisting a disaster scenario. *Future Generation Computer Systems* **90**, 129 – 148 (2019). <https://doi.org/https://doi.org/10.1016/j.future.2018.07.048>, <http://www.sciencedirect.com/science/article/pii/S0167739X18303649>
29. Trianni, V.: *Coordinated Motion*, pp. 73–95. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). [https://doi.org/10.1007/978-3-540-77612-3\\_6](https://doi.org/10.1007/978-3-540-77612-3_6), [https://doi.org/10.1007/978-3-540-77612-3\\_6](https://doi.org/10.1007/978-3-540-77612-3_6)
30. Varughese, J.C., Hornischer, H., Zahadat, P., Thenius, R., Wotawa, F., Schmickl, T.: A swarm design paradigm unifying swarm behaviors using minimalistic communication. *Bioinspiration & biomimetics* **15**(3), 036005 (2020)
31. Vásárhelyi, G., Virágh, C., Somorjai, G., Tarcai, N., Szörényi, T., Nepusz, T., Vicsek, T.: Outdoor flocking and formation flight with autonomous aerial robots. In: *2014 IEEE/RSJ Int. Conf. on Intell. Robots and Systems*. pp. 3866–3873 (2014). <https://doi.org/10.1109/IRoS.2014.6943105>
32. Wanninger, C., Eymüller, C., Hoffmann, A., Kosak, O., Reif, W.: Synthesising Capabilities for Collective Adaptive Systems from Self-Descriptive Hardware Devices - Bridging the Reality Gap. In: *8th Int. Symp. On Leveraging Appl. of Formal Methods, Verification and Validation* (Sept 2018)
33. Wanninger, C., Badem, T., Schörner, M., Eymueller, C., Poeppel, A., Reif, W.: Golive a modular mixed reality simulation for semantic plug and play. In: *2023 23rd International Conference on Control, Automation and Systems (ICCAS)*. pp. 1521–1525 (2023). <https://doi.org/10.23919/ICCAS59377.2023.10316758>
34. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering* **2015** (2015)