Let's do the swarm flight again: unleashing the potential of PROTEASE 2.0 for drone formation flight

Oliver Kosak, Philipp Kastenmüller, Wolfgang Reif

Angaben zur Veröffentlichung / Publication details:

Kosak, Oliver, Philipp Kastenmüller, and Wolfgang Reif. 2026. "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* 27 (2025) (6): 577–96. https://doi.org/10.1007/s10009-025-00834-w.

International Journal on Software Tools for Technology Transfer (2026) 27:577–596
https://doi.org/10.1007/s10009-025-00834-w

FOUNDATION FOR MASTERING CHANGE

Special Section: Rigorous Engineering of Collective Adaptive Systems

Let's do the swarm flight again: unleashing the potential of PROTEASE 2.0 for drone formation flight

Oliver Kosak[1] · **Philipp Kastenmüller**[1] · **Wolfgang Reif**[1]

Accepted: 13 December 2025 / Published online: 8 January 2026

Abstract

Drone formation flights, exemplified by performances such as the Intel Drone Shows, demonstrate the advancements and capabilities of current technology. This work revisits the concept of self-organization through swarm behavior for this goal, presenting PROTEASE 2.0 as an advanced approach in this domain. The proposed method facilitates parametrizable swarm behavior at a high level of abstraction. Building upon its predecessor, PROTEASE, it enables the generation of emergent effects through a single, generalized implementation, wherein only the parameters governing individual swarm members need to be adjusted. Leveraging swarm behavior for formation flight offers distinct advantages, including enhanced scalability, robustness, and flexibility. Unlike centrally coordinated approaches, swarm-based methods support the emergence of complex and dynamic formations. Notable formations include parallel swarms interacting with one another, single swarms utilizing multiple reference points to achieve novel flight patterns, and hierarchical swarm structures that further extend the range of possible configurations of swarm behavior. This paper introduces fundamental swarm behaviors that can be realized within the PROTEASE 2.0 framework in detail and explores their composition into more complex formations. The primary focus is the experimental and empirical evaluation of these concepts in simulated environments, including their stabilization properties when facing disturbances. In combination with previous successful pre-evaluations involving real drones it provides a strong foundation for future real-world applications of PROTEASE 2.0.

Keywords Swarms · Drones · Formation flight · Multi-robot systems

1 Introduction

Drone formation flights, e.g., demonstrated by the Intel Drone Shows [1], serve as a compelling demonstration of the latest advancements in technology. These performances involve the precise coordination of numerous drones, executing complex maneuvers as they navigate the night sky. Through synchronization, the drones form dynamic patterns that seamlessly integrate artistics with cutting-edge technological innovation. Their orchestrated movements offer a fusion of creativity and state-of-the-art engineering.

✉ O. Kosak
kosak@isse.de

P. Kastenmüller
philipp.kastenmueller@uni-a.de

W. Reif
reif@isse.de

[1] Institute for Software and Systems Engineering, University of Augsburg, Universitätsstraße 6a, Augsburg, 86152, Germany

In this work, we once again explore such application of multi-robot systems through the lens of self-organization, specifically employing swarm behavior. Utilizing swarm intelligence for formation flight offers significant advantages, including enhanced scalability, resilience, and adaptability. Unlike centrally controlled approaches, swarm-based coordination enables the creation of complex and visually striking formations while maintaining operational efficiency in dynamic environments. With our investigations in this article, we extend our contributions from [2], where we introduced PROTEASE 2.0. Our approach builds on a common pattern, we first identified in [3]: PROTEASE (an Algorithmic Pattern for Trajectory-Based Swarm Behavior).[1] This approach offers the possibility to dynamically adjust the parameters of a once implemented local behavior each swarm member executes, achieving adaptable swarm behavior on the macro level. This foundation allows for the creative recombination

[1] "... proteases are key regulators of a striking variety of biological processes ... they regulate different processes in response to developmental and environmental cue" [4].

of such swarm behavior in PROTEASE 2.0, where we can achieve fascinating effects like an *Olympic Rings* formation and others. To be more precise, PROTEASE 2.0 allows 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.

Through that flexibility, our approach differentiates itself from others: Following [5], a prevailing challenge in swarm application development is the tendency to create a distinct software solution for each different swarm behavior. Although these tailored approaches effectively fulfill their intended purposes, such as leveraging collective swarm behavior for search tasks [6] or distributed surveillance [7, 8], they often lack adaptability. As a result, users encounter difficulties in modifying these solutions for slightly different scenarios, thereby limiting their ability to build upon previous advancements. To overcome this challenge, we propose the adoption of our framework that more broadly captures the swarm behavior of a specific class. By incorporating this framework during the design phase, developers of aerial multi-robot systems can efficiently integrate swarm dynamics and later adjust its parameters at runtime to achieve diverse emergent behaviors.

While our primary focus here is on the visual aspects of the swarm behaviors, we believe that our approach could also improve the goal-oriented applications of swarm behavior in future developments. For instance, it may enhance the use of particle swarm optimization (PSO) in search and rescue missions, as discussed in [9]. In this work, we concentrate on the fundamental concepts and algorithm design within an abstract simulation environment. Our next logical step for future investigation is to integrate our findings with real-world drone formation flights. To this end, we already have an established method [3], which combines the multi-agent framework Jadex [10] for high-level decision-making with the Robotics Operating System (ROS) framework [11], along with the Gazebo physics simulation [12].

We will use that to actually establish the benefits we see in adopting swarm formation flight for real drones: Utilizing swarm algorithms for formation flight marks a significant shift in the control of autonomous aerial systems, offering advantages in scalability, robustness, and flexibility. By emulating natural phenomena such as bird flocks or fish schools, swarm algorithms enable autonomous vehicles to collaborate efficiently in the air [13]. This approach achieves coordination without the need for a central controller, promoting decentralized decision-making and ensuring real-time adaptability in dynamic environments. The fault tolerance inherent in swarm algorithms boosts reliability, which is essential for applications requiring high operational resilience. Additionally, the scalability of these algorithms facilitates the easy integration of additional units into the formation, a feature that could be valuable in other fields requiring spatial coordination, such as distributed surveillance [8, 14] or agriculture [15].

The contribution presented in this article, as an extension of [2], includes the following advancements: **(1)** A new schematic and formal description of basic swarm behaviors applicable to formation flight, such as LINE-OF-FLIERS and BALL-OF-FLIERS. **(2)** An expanded description of the potential emergent swarm behaviors that result from combining different basic swarm behaviors. **(3)** A broader range of experiments that explore the aforementioned basic and extended swarm behaviors including an empirical study concerning the self-adaptation properties of our approach. **(4)** A comprehensive account of our current efforts to implement the extended swarm behaviors using PROTEASE 2.0, with real drones in our flight arena.

To provide context, we begin by briefly introducing the concept of PROTEASE in Sect. 2. Next, we discuss related work on parametrizable swarm behavior in Sect. 3. In Sect. 4, we present a set of fundamental swarm algorithms that form the basis of our work, along with our novel approach to multi-layered, parametrizable swarm behavior, which we term PROTEASE 2.0. We evaluate our approach through a proof-of-concept demonstration in Sect. 5, supported by references to our source code and video materials available on GitHub. Finally, in Sect. 6, we summarize our findings and provide an outlook on potential future directions.

2 The idea of PROTEASE

PROTEASE is a framework for generalizing swarm behavior through a unified local control mechanism executed by each agent within a swarm. The underlying concept is to define a set of fundamental behavioral rules that govern individual swarm members while allowing external parameterization to achieve various swarm behaviors. This approach aims to standardize commonly employed functionalities in aerial robotic swarm systems, facilitating their implementation across diverse applications.

Despite exhibiting distinct emergent effects, many swarm algorithms – including Particle Swarm Optimization (PSO) [6], flocking behavior [16], formation control [17], and distribution strategies [7, 8] – share a core set of local actions. These actions primarily involve: measuring relevant parameters, exchanging information with neighboring agents, and adjusting the movement vectors of individual swarm members accordingly.

For instance, in the classical flocking behavior described by Reynolds [16], each "boid" executes specific functions such as position and velocity measurements, communication with other agents, and subsequent movement adjustments achieved through the weighted aggregation of so-

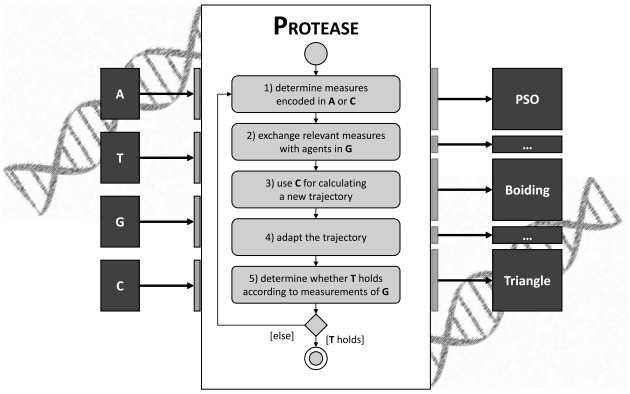


Fig. 1 The locally implemented behavior of PROTEASE each agent in a swarm executes

called urges. These local interactions collectively lead to emergent flocking behavior at the swarm level.

2.1 Parameterization of PROTEASE

Building upon these insights, we identified the essential parameters required to realize a broad spectrum of swarm behaviors within the PROTEASE framework [3] (cf. Fig. 1). The goal there was to integrate PROTEASE into higher-level task orchestration approaches, such as the one proposed in [18]. To this end, the behavioral model of PROTEASE is controlled by four key swarm functions:

- **Aggregation (A)** determines the collective result based on swarm interactions.
- **Termination (T)** defines the conditions under which the swarm considers its objective achieved.
- **Grouping (G)** establishes the local neighborhood for information exchange among agents.
- **Calculation (C)** specifies how information is processed to movement trajectories.

Each agent within the swarm executes these functions cyclically in predefined execution rounds. The process follows the structured sequence presented in Fig. 1:

1. Measure all relevant data required for A and C.
2. Exchange measurements with neighboring agents as defined by G.
3. Compute a new movement trajectory using C.
4. Adjust the movement trajectory accordingly.
5. Terminate execution if the termination criterion T is met, or otherwise, restart the cycle.

2.2 Examples of swarm behaviors

By appropriately configuring the (A T G C) parameters, various swarm behaviors can be realized. For instance, *Boid Flocking* [16] allows agents to align their movements based on the velocity and positional measurements of neighboring agents. *Triangle Formation* [8] enables the swarm to organize into triangular formations, which are particularly useful for observation and surveillance tasks. Additionally, *PSO-based Search* [6] allows the swarm to locate the highest concentration of a continuously distributed parameter P by evaluating centroids, geometric diameters, and local PSO rules. Furthermore, additional swarm behaviors can be formulated and executed using the (A T G C) parameter set, as further explored in [3].

2.3 Constraints and assumptions

To align with the PROTEASE framework, several fundamental assumptions must be satisfied. First, PROTEASE operates independently of environmental markings, unlike stigmergy-based algorithms such as ant-colony optimization [19], which rely on persistent modifications to the environment. This design choice is motivated by its intended application in aerial drone swarms [20], where implementing stigmergy would require a globally accessible digital representation of the environment.

Second, the primary focus of PROTEASE is on emergent swarm behaviors that arise from local trajectory adjustments. Swarm behaviors that depend on other agent functionalities are not effectively captured within this framework. Additionally, each agent is assumed to be capable of communicating with any other swarm member, which is particularly important for mechanisms such as self-termination, which are often used in PSO-based behaviors. Despite this, the actual execution of swarm rules remains localized, depending on the local environment defined by G.

Another key assumption is that agents may not have access to precise measurements of the positions and velocities of other agents in real-world applications, requiring them to rely on appropriate sensing techniques. Finally, all agents executing PROTEASE must operate in a three-dimensional space, which ensures compatibility with aerial robotic swarms.

3 Related work

The body of literature on swarm behavior, swarm algorithms, and swarm intelligence is both extensive and diverse. In the context of real-world applications, research in this field generally follows two primary methodological approaches.

Specific swarm behavior One category of approaches focuses on the analysis and adaptation of specific swarm behaviors for implementation in technical systems. While numerous studies have explored this methodology, only a selection of relevant research can be discussed here.

For instance, particle swarm optimization (PSO) has been widely applied to various search problems, encompassing both real-world implementations with flying robots [21, 22] and simulations [23, 24]. Additionally, researchers have drawn inspiration from the self-coordination mechanisms observed in swarms of birds and fish, integrating these principles into technical systems to minimize coordination efforts, particularly when managing large groups of aerial robots [13, 21, 22].

Several studies have also explored the development of software controllers for individual swarm agents using genetic algorithms and other learning techniques. Such methods have been employed for coordinated motion [25], foraging tasks [26], and combined search-and-action tasks [27]. Moreover, some approaches rely on meticulously engineered, situation-specific solutions, such as the formation of geometric patterns on the ground using problem-specific robotic platforms like Kilobots [17].

In the domain of distributed surveillance, Ma et al. [7] proposed a deployment algorithm based on potential fields to achieve a nearly uniform distribution of swarm entities within a designated area. However, the applicability of this algorithm is limited to its specific use case. Li et al. [8] extended their swarm approach for distributed surveillance to include flocking and obstacle avoidance, yet they did not further explore its generalization. This extension represents a promising step toward establishing a generalized pattern for swarm behavior, aligning with our approach.

Further examples include Garcia et al. [21], who modified the PSO algorithm for deploying flying robots in disaster scenarios to facilitate area exploration and victim detection. Despite the ability to adjust parameters to accommodate different objectives, this approach remains constrained by its narrowly defined application scope and lacks generalizability. Similarly, Vasarhelyi et al. [13] adapted the flocking algorithm described in [16] to enable flying robots to replicate natural swarm behavior. However, their implementation is highly specific and does not allow for the realization of other swarm behaviors beyond the predefined scope.

Lastly, Dedousis et al. [5] introduced swarm primitives designed to manage multi-robot systems through their PaROS approach. While their work focuses on implementing swarm behavior for specific tasks using these primitives, it does not extend to the generalization of swarm behavior as a whole.

Generalizing swarm behavior Another category of approaches focuses on developing generalized frameworks for collective behavior rather than tailoring solutions to specific applications. These frameworks provide adaptable methodologies that can be applied to a variety of scenarios and requirements.

One example is Protelis [28] and its successor MacroSwarm [29], which conceptualize entities within a collective system as components of vector fields. This approach enables collective programming by manipulating operations within these fields, with changes propagating through implicit communication among entities. This abstraction allows for the implementation of complex collective behaviors. However, Protelis presents challenges in prototyping 3D swarm behavior due to limited integration with simulation frameworks and drone hardware. Additionally, it lacks support for all features of PROTEASE 2.0, particularly hierarchical swarm structures.

Similarly, Buzz [30] serves as a framework for generalizing and programming collective behavior. In Buzz, swarms function as primary abstractions, with tasks dynamically distributed among individual robots through local interactions. The framework employs virtual stigmergy, utilizing distributed tuple spaces to achieve swarm-wide consensus, and allows for extensibility via new primitives. However, it does not support hierarchical swarm structures, a key feature of PROTEASE 2.0.

Another approach, Meld, introduced by Rollman et al. [31], manages swarm behavior through high-level abstractions. However, its execution model is not well-suited for real-world aerial swarms, as ensemble behavior computations require significant processing time. Additionally, the constrained computational resources of flying robots present further challenges to practical implementation, making Meld's execution model more limited compared to our approach.

Lastly, Varughese et al. [32] propose a design paradigm aimed at generalizing swarm behavior by minimizing communication requirements. Their method emphasizes decentralized and minimalistic functionality, suggesting that complex behaviors, such as collective transport in 2D environments, can emerge from combinations of simple primitives. However, their approach relies on cyclic synchronization messages, leading to extended stabilization times (e.g., 1400 seconds for distributed localization). This characteristic may limit its effectiveness for flying swarms, where real-time adaptability is crucial.

Formation flight and robot control When widening the horizon for the application of PROTEASE 2.0 for real-world execution, there exists a body of research focused on formation flight and robot control that intersects with our work. For instance, Bettini et al. [33] provides a programming methodology for distributed multi-robot systems integrating the Robot Operating System (ROS) [11] with X-KLAIM, enabling a simplified interface to programming heterogeneous robots' behaviors. While the approach could possibly be adapted to realize hierarchic behavior of multi-robot systems, it currently lacks proof-of-concept for controlling swarms or high-scale multi-robot systems, respectively which probably could be improved by using a more up to date version of ROS, e.g., ROS2 [34].

Another approach enabling programming of distributed multi-robot systems dedicated to formation flight (DROONA) is presented by Desai et al. [35]. The approach allows for provably correct planned formation flight trajectories for multi-drone systems, achieved in a decentralized fashion. However, the approach aims for drones to reach predefined goal positions for each drone it plans trajectories for—which we aim to avoid in our approach PROTEASE 2.0 by using the paradigm of self-organization when applying swarm behavior to drone formation flight. Further, DROONA in its current state does not yet prove to be highly scalable, as it is a goal for swarm robotic systems.

A further approach for realizing formation flight is given by Koord [36], which is able to verify that drone trajectories for multi-robot systems follow predefined goals. In contrast to our work focusing on the paradigm of self-organization (we support up to 384 in this paper), Koord struggles with higher agent numbers (up to 15 robots) as it aims for verified correct programs. Swarm Robotics instead inherently takes into account being sub-optimal or even temporarily faulty but aims for providing robustness to stabilize again after failures without the need for replanning or user interaction.

Besides approaches from that literature, we already aimed for realizing formation flight with drones ourselves. In [37] we initially realized a first formation flight in a field experiment, which we then extended with an multi-agent based controlling framework in [38, 39] for geographical measurements. In [40] and [3, 18, 41] we then presented systematic approaches to generate greater mission context for formation flight with drones using configurable hardware modules [20, 42], enabling us also to involve goal-oriented swarm and formation flight, e.g., in major catastrophe scenarios, and made approaches for their safe operation [43].

4 Approach

The PROTEASE 2.0 approach extends PROTEASE by incorporating the concepts of multiple reference points. While in PROTEASE only one reference point could be used to externally guide a swarm through space, now multiple such reference points can be used. Moreover, swarm agents themselves can serve as reference points, leading to even more possibilities of combinable swarm behavior. This extension enables the creation of a multi-layered, parametrizable swarm behavior. Additionally, PROTEASE 2.0 distinguishes between configurations that utilize a single set of agents and those that employ multiple sets of agents.

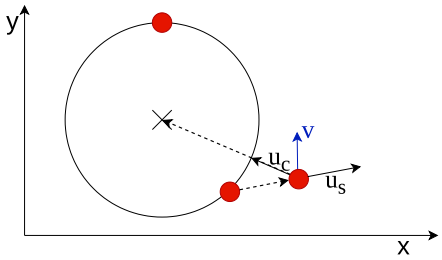
In this study, our primary focus is on the visual effects of swarm behavior. Consequently, we select the parameters for PROTEASE 2.0 accordingly. Specifically, for the extensions discussed in this paper, we do not consider the auto-termination mechanism of PROTEASE, which is typically employed for mission integration [3, 18]. As a result, both the aggregation function (A) and the termination function (T) are omitted from our current analysis. For the grouping function (G), we define the set of neighboring agents as those within a predefined radius around a given agent. While the calculation function (C) remains generally configurable, we limit our discussion to a specific excerpt of calculation functions in here: The Ring-of-Fliers, the Line-of-Fliers, and the Ball-of-Fliers functions that produce a ring, line, or ball swarm behavior.

4.1 Basic swarm algorithms

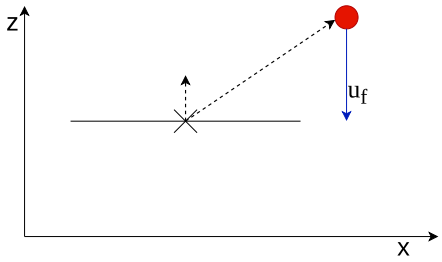
Not all possible swarm behavior that can be achieved with PROTEASE is applicable for the purpose of formation flight. Flocking behaviour according to [16], Particle Swarm Optimization behavior [21] and similar does not produce visually stable flight patterns. Thus, in the following we describe different swarm behavior, we can use therefore.

For each of these swarm behaviors, every agent within the swarm is provided with the necessary input information. This includes its own position, the six-dimensional transformation of reference point defining the center of the desired formation, and possible swarm-behavior-specific parameters. The reference point may be implemented as a virtual element in simulation environments or as a physical device manipulated in real-world scenarios, such as an additional user-controlled drone. Additional information required for executing the swarm behavior, particularly data concerning an agent's neighboring swarm members, is obtained automatically through the multicast-like communication process occurring among all neighbors within G (cf. Fig. 1). Each swarm behavior emerges from a combination of local urges, following the swarm model introduced by [16]. This model integrates multiple behavioral urges, such as cohesion, separation, and alignment, to determine the agent's next trajectory in three-dimensional space.

In the following, the expressions c_{OWN} and c_{REF} denote the agents' capabilities c for measuring their OWN six-dimensional transformation and that of the reference point REF, respectively. Both functions return a six-dimensional vector. In the following descriptions and equations, the methods POS() and ROT() extract the position and rotation (expressed as a unit vector each) from such a six-dimensional transformation. Additionally, CLOSE() determines the transformation to the closest agent within G, based on Euclidean distance. The operator $\circ$ represents the scalar product of vectors. For the illustrative purposes withing our figures, we employ a coordinate system in which the z-axis corresponds to the rotation of the reference point (cf. Figs. 2 to 5). However, it is possible to freely place a reference point and thus its associated shape produced by the respective swarm behavior in 3-dimensional space, as explained in the following section.



(a) Center and separation urges u_c and u_s for the `Ring-of-Fliers` basic swarm behavior.



(b) Flat urge u_f for the `Ring-of-Fliers` basic swarm behavior.

Fig. 2 Schematic for calculating urges in $\mathtt{C}_{\mathtt{ring}}$, i.e., the `Ring-of-Fliers` basic swarm behavior. 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

4.1.1 The `Ring-of-Fliers` behavior

To establish a ring formation, we define a computational function, $\mathtt{C}_{\mathtt{ring}}$, which governs the swarm behavior as detailed in Eqs. (1) to (7). The swarm-behavior-specific parameter r determines the radius of the ring. Each agent within the swarm is provided with the necessary input data required for $\mathtt{C}_{\mathtt{ring}}$ to facilitate formation execution.

The ring formation is achieved through the localized combination of three distinct urges, u_c, u_s, and u_f for each agent (see Eqs. (2) to (4)). Within the computation of the `Ring-of-Fliers` pattern, each urge is represented as a vector, indicating the agent's desired trajectory. The center urge u_c directs the agent toward the measured position representing the center of the ring, i.e., the position of the reference point. The separation urge u_s repels the agent from the nearest identified neighbor, as determined by applying the function CLOSE on the agent set returned by the grouping function G. The flat urge u_f guides the agent towards the closest point on the plane defined by the reference point and its associated normal vector. The function $\mathtt{C}_{\mathtt{ring}}$ subsequently computes the resultant trajectory $v_{\mathtt{ring}}$ by applying a weighted sum of these urges, incorporating the respective weighting coefficients ω_c, ω_s, and ω_f, thereby determining the new movement vector for each agent:

$$v_{\mathtt{ring}} := \omega_c \cdot u_c + \omega_s \cdot u_s + \omega_f \cdot u_f \tag{1}$$

$$u_c := \mathtt{POS}(c_{\mathtt{REF}}) - \mathtt{POS}(c_{\mathtt{OWN}}) \tag{2}$$

$$u_s := \mathtt{POS}(c_{\mathtt{OWN}}) - \mathtt{POS}(\mathtt{CLOSE}(\mathtt{G})) \tag{3}$$

$$u_f := \mathtt{ROT}(c_{\mathtt{REF}}) \tag{4}$$

$$\omega_c := \frac{|u_c| - r_{\mathtt{RING}}}{|u_c|} \tag{5}$$

$$\omega_s := \frac{1}{|u_s|^2} \tag{6}$$

$$\omega_f := (\mathtt{POS}(c_{\mathtt{REF}}) - \mathtt{POS}(c_{\mathtt{OWN}})) \circ \mathtt{ROT}(c_{\mathtt{REF}}) \tag{7}$$

4.1.2 The `Line-of-Fliers` behavior

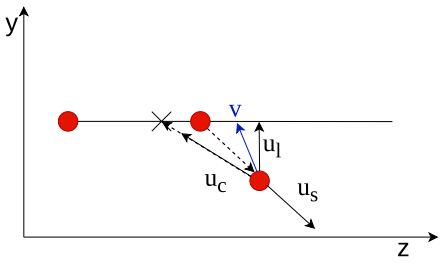
Similar to the `Ring-of-Fliers`, we can define a calculation function $\mathtt{C}_{\mathtt{line}}$ producing a line-like swarm behavior (cf. Eqs. (8) to (11)). Each agent participating in the swarm is provided with the required information as an input for $\mathtt{C}_{\mathtt{line}}$. The swarm-behavior-specific parameter $r_{\mathtt{line}}$ here determines the length of the line. The line formation then establishes through the local combination of three different urges u_c, u_s, and u_l for each agent (cf. Eqs. (2), (3) and (9) to (11)). In the calculation for the `Line-of-Fliers` pattern described here, each urge is represented by a vector indicating the desired trajectory for an agent. The center urge u_c as well as the separation urge u_s are equivalent to the urges described in Sect. 4.1.1 (cf. Eqs. (2) and (3)). The line urge u_l points towards the closest point on the line defined by the orientation of the reference point. The function $\mathtt{C}_{\mathtt{line}}$ then calculates the resulting trajectory $v_{\mathtt{line}}$ using the weighted sum of the urges with the respective weights ω_c and ω_s for a new agent movement vector v. The calculation of the weighting coefficients ω_c and ω_s is identical to the approach employed for the `Ring-of-Fliers` (cf. Eqs. (5) and (6)), while ω_c of course uses $r_{\mathtt{line}}$ here.

$$v_{\mathtt{line}} := \omega_c \cdot u_c + \omega_s \cdot u_s + u_l \tag{8}$$

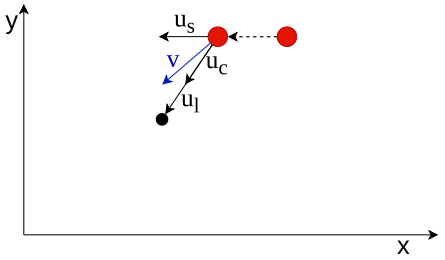
$$u_l := proj - \mathtt{POS}(c_{\mathtt{OWN}}) \tag{9}$$

$$proj := \mathtt{POS}(c_{\mathtt{REF}}) + c_p \cdot \mathtt{ROT}(c_{\mathtt{REF}}) \tag{10}$$

$$c_p := \frac{(\mathtt{POS}(c_{\mathtt{OWN}}) - \mathtt{POS}(c_{\mathtt{REF}})) \circ \mathtt{ROT}(c_{\mathtt{REF}})}{\mathtt{ROT}(c_{\mathtt{REF}}) \circ \mathtt{ROT}(c_{\mathtt{REF}})} \tag{11}$$



(a) Center, separation and line urges (u_c, u_s and u_l) for the `Line-of-Fliers` swarm behavior



(b) The line urge u_l illustrated with the line oriented towards the reader.

Fig. 3 Schematic for calculating urges in $\mathsf{C}_{\texttt{line}}$, i.e., the `Line-of-Fliers` basic swarm behavior. Dashed arrows represent measured values for one agent in the red swarm. Solid arrows show calculated urges u_c, u_s, and u_l. The blue arrow indicates the resulting trajectory v

4.1.3 The `Ball-of-Fliers` behavior

To establish a ball formation, we define the computational function $\mathsf{C}_{\texttt{ball}}$, which generates the desired swarm behavior. The swarm-behavior-specific parameter $r_{\texttt{ball}}$ determines the radius of the ball. The `Ball-of-Fliers` algorithm closely resembles the `Ring-of-Fliers` algorithm, with the primary distinction being the omission of the flat urge. The ball formation emerges from the localized combination of two distinct urges for each agent, u_c and u_s (cf. Eqs. (2) and (3)). The function $\mathsf{C}_{\texttt{ball}}$ determines the resulting trajectory v_{ball} by computing a weighted sum of these urges, incorporating the respective weighting coefficients ω_c and ω_s (cf. Eqs. (5) and (6)), where ω_c uses $r_{\texttt{ball}}$ here. The new agent movement vector is thus given by

$$v_{ball} := \omega_c \cdot u_c + \omega_s \cdot u_s \tag{12}$$

4.2 Multiple reference points

While the cyclic 'default' computation model for PROTEASE, as established in our previous findings [3, 9] is detailed in Sect. 2 and applied to formation flight swarm behavior in Sect. 4.1, this section extends the PROTEASE framework to incorporate multiple reference points simultaneously. This extension enables the formation of novel ring-, line-, and ball-based flight patterns, which we schematically illustrate in Fig. 4. Despite this extension, all agents continue to execute their calculations in discrete execution rounds asynchronously, which persist until an independent termination signal is received.

Algorithm 1 execution of c^*_{REF} in C^* integrated with the CLOSE selection strategy

1: REF = CLOSE(REF)
2: **RETURN** c_{REF}

1) Single agent set approach By generalizing the results of c_{REF}, which determines the 6-D transformation of the reference point, from a single measured value REF to a set of measurements REF, each agent must now select one of these values when computing urges associated with the reference point, such as u_c and u_f. While alternative selection strategies exist, we adopt an approach that prioritizes the reference point closest to the agent's current position when executing c_{REF}, thereby extending c_{REF} to c^*_{REF} and thus C to C^* (cf. Algorithm 1). Since the selection process in Algorithm 1 is updated during each execution cycle of PROTEASE 2.0 (cf. the execution model in Sect. 2), agents may dynamically adjust their reference point selection in every iteration. To implement this, we utilize CLOSE(REF) within each agent's C^* to determine the transformation of the nearest reference point in terms of Euclidean distance. Consequently, the reference point deemed relevant by each agent may shift over the course of PROTEASE 2.0 execution, particularly in response to user or environmental modifications. To maintain the intended behavior and minimize collision risks within the swarm, the separation urge u_s remains unchanged, ensuring that agents continue to move away from their closest neighbor within G independent of the neighbors associated reference point. Figure 4a illustrates the updated calculation of C^* for the `Ring-of-Fliers` base swarm behavior in scenarios involving two distinct reference points.

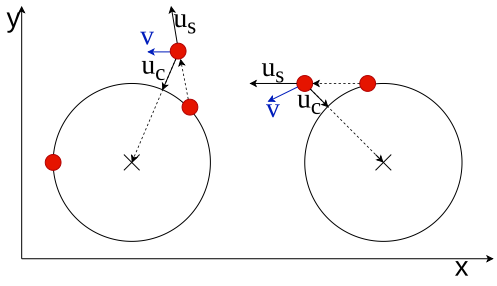
2) Multi agent set approach As a second adaptation of the computational function C incorporating multiple reference points, denoted as C^+, we explicitly assign each agent to a specific (randomly chosen) reference point when initializing the swarm, thereby restricting its local view to this designated reference point REF throughout the entire execution of PROTEASE 2.0 (cf. Algorithm 2). Thus, instead of allowing each agent to select a reference point dynamically from REF, we enforce a fixed assignment that remains consistent when computing reference-related urges using c^+_{REF}, such as u_c, u_f or u_l, across all execution rounds of PROTEASE 2.0 (cf. Algorithm 3). Even in cases where reference points shift positions over time, each agent continues to consider only its

Algorithm 2 random reference point selection strategy in $\mathtt{C}^+$ when initializing the swarm

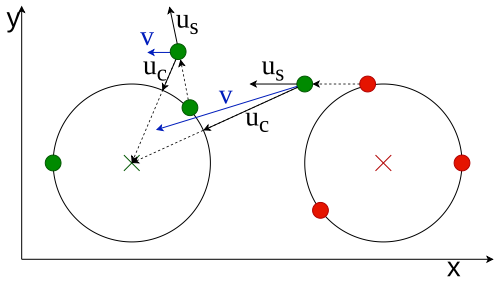
1: REF = `oneof` REF

Algorithm 3 execution of c^+_{REF} in $\mathtt{C}^+$

1: RETURN c_{REF}



(a) Single Agent Set $\mathtt{C}^*_{\mathtt{ring}}$



(b) Multi Agent Set $\mathtt{C}^+_{\mathtt{ring}}$

Fig. 4 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 $\mathtt{C}^+_{\mathtt{ring}}$ and $\mathtt{C}^*_{\mathtt{ring}}$. In Fig. 4b, green agents are assigned to the green reference point, while red agents are assigned to the red one. In Fig. 4a, all agents belong to the same agent set

assigned reference point, disregarding potentially closer alternatives. Analogous to $\mathtt{C}^*$ in the Single Set Approach, the separation urge u_s remains unchanged to maintain proper swarm dynamics. Figure 4b illustrates the modified computation of $\mathtt{C}^+$ for the `Ring-of-Fliers` base swarm behavior when two distinct reference points exist, and agents are explicitly assigned to reference points.

4.3 Hierarchies

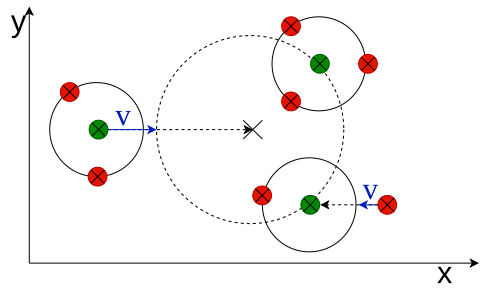
Beyond the extension with multiple reference points, we introduce a hierarchical structure within swarms by incorporating an adapted calculation function, denoted as $\mathtt{C}^{\mathtt{H}}$. In contrast to $\mathtt{C}$, $\mathtt{C}^*$, and $\mathtt{C}^+$, where reference points are externally controlled, $\mathtt{C}^{\mathtt{H}}$ extends the set of reference points to include all swarm agents themselves. Consequently, agents may be assigned to reference points that correspond not only to user-controlled devices but also to other agents within the swarm. This hierarchical organization facilitates the formation of structured sub-swarms by appropriately allocating agents to reference points. To maintain system stability and prevent unintended behaviors, reflexive allocations must be strictly avoided, and cyclic dependencies must be carefully regulated, as improper assignments may lead to chaotic swarm dynamics. Within this hierarchical framework, each layer comprises a set of agents that collectively contribute to the emergent swarm behavior. At the first hierarchical level, designated as `layer` 0, a user-controlled reference point governs the entire system. In subsequent hierarchical levels, denoted as `layer` i, where $i \in \{1..n\}$, agents are assigned reference points from the preceding layer, `layer` $(i-1)$. Thus, each agent $\alpha_x \in \mathcal{A}_i$ in `layer` i (where $i > 0$) utilizes an agent $\alpha_y \in \mathcal{A}_{i-1}$ as its reference point, where $\mathcal{A}_i$ represents the set of agents at hierarchical level i and $\mathcal{A}_{i-1}$ those of layer $i-1$ respectively. This structured approach enables scalable and adaptive swarm behaviors while preserving coherence across hierarchical levels.

The assignment of reference points to agents can be implemented in a fixed manner, analogous to $\mathtt{C}^+$, which we denote as $\mathtt{C}^{H+}$. This approach results in stable hierarchical structures by enforcing predefined reference point allocations. Alternatively, similar to $\mathtt{C}^*$, the selection of a reference point can be delegated to the agents themselves, represented as $\mathtt{C}^{\mathtt{H}*}$. In this case, only the assignment of agents to hierarchical layers is predefined, allowing for increased system autonomy while reducing the extent of direct control over emergent visual patterns at the collective level.

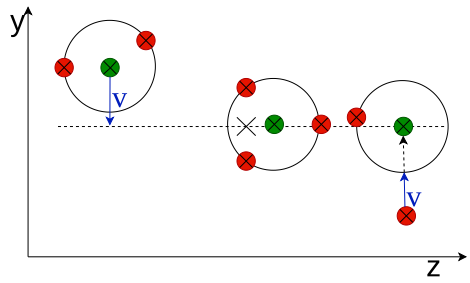
An additional feature of hierarchical swarm structures is the capability of reference points to provide distinct computational functions $\mathtt{C}$. Examples include $\mathtt{C}_{\mathtt{ring}}$, $\mathtt{C}_{\mathtt{line}}$, and $\mathtt{C}_{\mathtt{ball}}$ as described in Sect. 4.1 among others, thereby expanding the range of possible collective behaviors. Consequently, various swarm formations can be combined hierarchically, and individual agents may dynamically adjust their behavior based on their currently assigned reference point. This assignment can also change over time, depending on the computational function associated with the reference point. The hierarchical execution illustrated in Fig. 5a demonstrates a two-layered structure where $\mathtt{C}_{\mathtt{ring}}$ is applied uniformly across both hierarchical layers, `layer 0` and `layer 1` which we ascribe as $\mathtt{C}^{\mathtt{H}+}_{\mathtt{ring},\mathtt{ring}}$. In Fig. 5b, we showcase the possibility of mixing different behaviors on different layer of the swarm, i.e., $\mathtt{C}^{\mathtt{H}+}_{\mathtt{line},\mathtt{ring}}$. The depicted formation establishes when we assume a line swarm behavior on `layer 0` and a ring behavior on `layer 1`.

4.4 Adaptation to changes

One of the main features we want to make profit of when using a swarm of agents is its adaptivity to changes in



(a) Hierarchic concept of PROTEASE 2.0 using the C_{ring} parameter on each layer, structured as a 2-layered Ring-of-Rings.



(b) Hierarchic concept of PROTEASE 2.0 using the C_{line} on layer 0 and the C_{ring} parameter on layer 1, structured as a 2-layered Line-of-Rings.

Fig. 5 Agents with crosses can serve as reference points in C^H. Green agents (`layer 0`) use the user controlled reference point, whereas red agents (`layer 1`) use green agents as their reference points. Only the resulting trajectory v (v_{ring} or v_{line} respectively) is shown

the environment or the user interaction. Thus, we want to briefly discuss how all parametrizations of PROTEASE and thus PROTEASE 2.0 can handle different changes during execution.

Obstacle avoidance To avoid collisions with obstacles in the environment, we can add an additional urge u_o to each of the calculation functions C described above that acts similar to the seperation urge u_s. Instead of 'away from the closest neighbor', the urge points 'away from the closest obstacle' within a certain perception-range. The resulting trajectory v then is calculated by adding this urge to the respective equations, e.g., for the `Ring-of-Fliers` in Eq. (13) or for the `Line-of-Fliers` in Eq. (14), where ω_o is the weight of the obstacle avoidance urge.

$$v_{ring} := \omega_c \cdot u_c + \omega_s \cdot u_s + \omega_f \cdot u_f + \omega_o \cdot u_o \tag{13}$$

$$v_{line} := \omega_c \cdot u_c + \omega_s \cdot u_s + u_l + \omega_o \cdot u_o \tag{14}$$

Agent removal When an agent is removed from the swarm, the swarm as a whole adapts to this change. Because each agent only uses local information, the remaining agents either do not need to adapt as their neighborhood is undtouched by the removal or they automatically adapt wihtin the next execution round of their local rules. For example, in the `Ring-of-Fliers` behavior, the separation urge u_s changes for the agents that had the removed agent as their closest neighbor. Thus, the agents automatically move closer together beacuse the seperation urge is no more influenced by the now removed agent that previously pushed those agents away from its positon. This, as an emergent effect, automatically fills the gap left by the removed agent in a self-organized way.

Agent integration When a new agent is added to the swarm, it integrates smoothly into existing, probably complex, formations. Therefore, it first discovers its respective reference point according to the current reference point selection strategy (Multi- or Single Set Approach), i.e., a random or a predefined reference point. Then, the new agent starts measuring the position of that reference point as well as those of the closest neighbors in its perception range. It then calculates its urges based on these measurements, bringing the agent closer to the formation and eventually into formation. As soon as the new agent comes into the perception range of other agents, these agents also adapt their urges based on the position of the new agent. For example, in the `Ring-of-Fliers` behavior, the new agent calculates its separation urge u_s based on the positions of its closest neighbors, while these agents also update their separation urge u_s based on the position of the new agent. Thus, the new agent automatically moves to a position within the desired formation (enforced by the flat urge u_f and the center urge u_c) while the other agents get pushed away slightly to make space for the new agent (enforced by the seperation urge).

Conditional behavior and transitions Because we allow agents in PROTEASE 2.0 to change not only single parameters of their calculation function but to also exchange the whole set of parameters with respect to the reference point they are the closest to (cf., e.g., in C^*), agents can also change their respectively executed behavior in other conditions in a self-adaptive manner. For example, agents can switch between the `Ring-of-Fliers` and the `Line-of-Fliers` behavior based on their distance to other designated positions. It is evident that we can also modify the agents' behavior in C^+ and C^H when we change the parameters of the reference points or the assignment of agents to reference points externally. For every situation we change one or more parameters, the agents' receiving the update instantly adjust the calculation of their urges and thus their resulting trajectory for the next execution round of PROTEASE 2.0 (i.e., their local rules). Because all calulation functions are linear combinations of separate urges, the transition between different behaviors is

smooth and continuous. This effect is even intesified because of the fact that the separation urge u_s is represented in all calculation functions (as well as the obstacle avoidance urge u_o, if active). That way, we can obviously also move the whole swarm from one position to another by changing the position of the reference point smoothly, because all agents will follow the reference point while still maintaining their formation.

5 Evaluation

The general applicability of PROTEASE in its original form was previously validated in [3]. In that study, PROTEASE was analyzed by employing comparable parameters A, T, G, and C to generate a diverse range of emergent behaviors, thereby inducing corresponding modifications in swarm dynamics. In the present evaluation, we extend this analysis to assess the enhanced concepts introduced in PROTEASE 2.0, specifically its ability to incorporate multiple reference points and facilitate hierarchical swarm structuring, as proposed in [2]. To conduct these assessments, we primarily utilize the NetLogo multi-agent programmable modeling environment.[2] On the one hand, our experimental setting covers multiple proof-of-concept experiments where we showcase the result of executing PROTEASE 2.0 with variable sets of parameters, each producing different visual effects (cf. Sects. 4.1 to 4.3) and analyze in which situation which parametrization should be used for achieving the desired goal. On the other hand, we also provide an empirical evaluation of the adaptivity properties PROTEASE 2.0 provides (cf. Sect. 4.4). Therefore, we quantitatively analyze the behaviuor of representative parameter sets for PROTEASE 2.0, i.e., basic swarm behavior, multi-swarm beahavior, as well as hierarchic swarm behavior. We stress each configuration with system-internal disturbances in the form of integrating new agents (scalability), removing agents (robustness), modifying goals (flexibility), and disturbances introduced by the environment in the form of spontanuous occuring obstacles (general adaptivity). Additionally, we provide initial video materials demonstrating the application of these concepts to real-world drone systems.

5.1 Evaluation environment

NetLogo [44] is a dedicated programming environment designed to facilitate the simulation and modeling of multi-agent systems. It abstracts many of the complexities associated with agent-based simulations by employing appropriate discretization techniques, thereby lowering the barriers to prototyping collective and swarm behaviors. NetLogo enables the simulation of agents within both two- and three-dimensional environments while incorporating a logical time model to govern state transitions. Within this environment, agents are spatially positioned and oriented, allowing them to perceive environmental data and utilize it for internal computations. Furthermore, agents are capable of bidirectional communication, enabling the unrestricted exchange of information. The simulation employs a cyclic execution mechanism, ensuring that agents perform their computations in a round-based manner. In our visual representations, agents are depicted as arrows, illustrating both their position and orientation in the three-dimensional environment. The color of these arrows conveys additional information, such as the agent subset or the hierarchical layer to which they belong.

To evaluate the concepts introduced in Sect. 4, we utilize two distinct NetLogo models, which we have made available on GitHub.[3] The first model, *MultiInput*, is designed to assess the extension incorporating multiple reference points, while the second model, *Hierarchic*, facilitates the evaluation of hierarchical swarm execution. The *MultiInput* interface allows users to specify the initial number of agents and reference points. Although our conceptual approach imposes no intrinsic limitations on the number of reference points, the current implementation supports a maximum of five. These reference points can be manipulated individually or through predefined movement presets. Additionally, the model allows for the modification of agent-specific parameters, including maximum velocity (`max-velocity`) and communication radius (`com-radius`). Users can further refine swarm behavior by adjusting parameters such as the radius of the RING-OF-FLYERS formation. Furthermore, they can choose between a homogeneous agent distribution or an initial assignment of agents to reference points. The *Hierarchic* model offers similar functionality while introducing additional capabilities for specifying distinct swarm behaviors at different hierarchical layers. Users can define swarm types, allocate agents to hierarchy levels, and configure communication radii. While our theoretical framework supports an arbitrary number of hierarchical layers, the implemented model imposes a practical limit to ensure computational feasibility.

5.2 Experimental proof-of-concepts

We provide several example configurations for the NetLogo implementation of PROTEASE 2.0. While these configurations already demonstrate intriguing complex swarm behaviors, we encourage readers to run their own experiments using the code available on GitHub in order to explore additional emergent effects resulting from the interaction of

[2] NetLogo download available at https://ccl.northwestern.edu/netlogo/download.shtml.

[3] PROTEASE 2.0 on GitHub at https://github.com/OliverISSE/Protease2.0.

Table 1 Comprehensive set of parameters for the first (Flower Pattern), the second (Olympic Rings Pattern), and the third (Letters *A* and *V* Pattern) experiments

Experiment	# agent	com-radius	max-velocity	function	$r_{\texttt{ring}}$, resp. $r_{\texttt{line}}$
flower-multi/-single	45	16	0.2	$\mathsf{C}^{*}_{\texttt{ring}}/\mathbf{C}^{+}_{\texttt{ring}}$	6
olympic-multi/-single	50	16	0.2	$\mathsf{C}^{*}_{\texttt{ring}}/\mathsf{C}^{+}_{\texttt{ring}}$	5
letter-V-multi/-single	50	6	0.2	$\mathsf{C}^{*}_{\texttt{line}}/\mathsf{C}^{+}_{\texttt{line}}$	18
letter-A-multi/-single	75	6	0.2	$\mathsf{C}^{*}_{\texttt{line}}/\mathsf{C}^{+}_{\texttt{line}}$	18

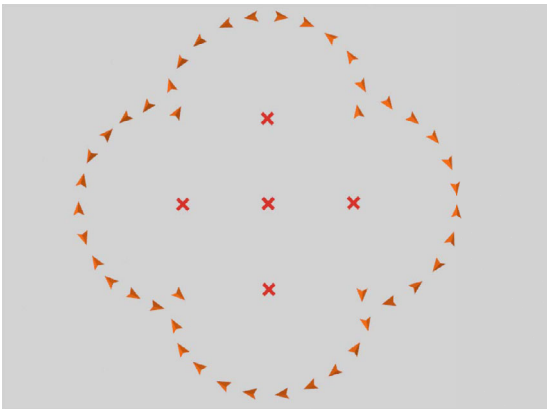
various swarm behaviors. Our experiments in this paper do not specifically investigate the typical swarm properties of *robustness* and *scalability*. However, these properties are inherent to PROTEASE 2.0, unless their excessive application diminishes the visual effects emphasized in this study. To further validate the general functionality of the model, we have extended our GitHub implementation to include the relevant features. Although the figures presented here primarily offer a 2-dimensional view of the swarm behavior (with the exception of Fig. 10), all experiments were conducted within the provided 3-dimensional environment. We observed significant qualitative differences in the swarm effects produced, and to explore these differences, we compare the emergent behaviors of the basic swarm calculation functions, $\mathsf{C}_{\texttt{ring}}$, $\mathsf{C}_{\texttt{line}}$, and $\mathsf{C}_{\texttt{ball}}$ (cf. Sect. 4.1). These are then combined for more complex executions using the Single Set Approach C^{*} and the Multi Set Approach C^{+} (cf. Sect. 4.2), as well as their integration with the Hierarchic Approaches $\mathsf{C}^{\mathrm{H}*}$ and $\mathsf{C}^{\mathrm{H}+}$ (cf. Sect. 4.3).

We provide the comprehensive set of parameters for each of the following experiments in Tables 1 and 2, allowing for the reproduction of the experiment using our implementation provided on GitHub.

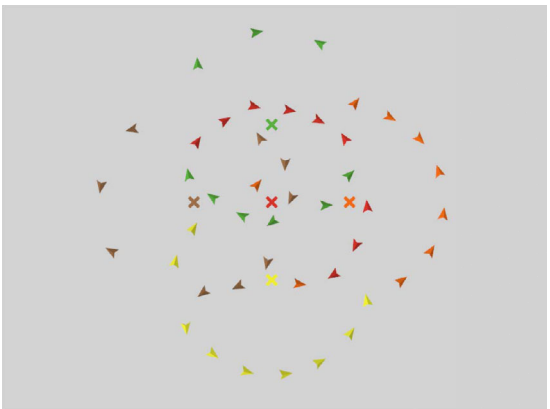
5.2.1 The flower pattern

In the first experiment using multiple reference points, we first utilize the Single Agent Set Approach $\mathsf{C}^{*}_{\texttt{ring}}$. We align five reference points in a cross-like formation. For each reference point, we decide on the `Ring-of-Fliers` swarm behavior using the Single Agent Set Approach and assure, that the distance between the reference point located in the middle of the structure to each other reference point is lower than $r_{\texttt{ring}}$. Due to the agents' behavior of always choosing the closest reference point, the rings don't close like assumed for basic swarm behavior. Instead, the result is a flower-like shape, where agents evenly distribute on the edge line of the shape (cf. Fig. 6a).

Switching to the Multi Agent Set Approach using $C^{+}_{\texttt{ring}}$ while leaving all other parameters untouched, forces the agents to create one ring for each reference point. Then, all rings close – also within the shape. Thus, we can observe their interference, which obfuscates the clear flower-like shape (cf. Fig. 6b). For experiment parameters see Table 1.



(a) Single Agent Set Approach ($\mathsf{C}^{*}_{\texttt{ring}}$)

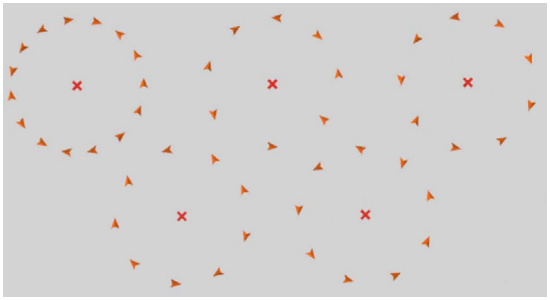


(b) Multi Agent Set Approach ($\mathsf{C}^{+}_{\texttt{ring}}$)

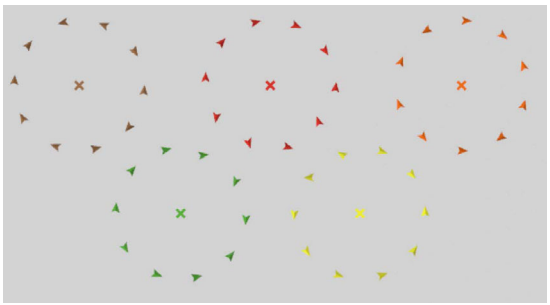
Fig. 6 Flower Pattern experiment producing a flower-like visual effect

5.2.2 The olympic rings pattern

In a second experiment using multiple reference points, we first utilize the Single Agent Set Approach. We align five reference points in a W-like formation where all reference points use the `Ring-of-Fliers` swarm behavior, i.e., $\mathsf{C}^{*}_{\texttt{ring}}$. If we also keep the distance between the reference points greater than $2r_{\texttt{ring}}$, we can observe the swarm creating a pattern similar to the famous Olympic Rings (cf. Fig. 7a). Agents do not evenly distribute across the different possible reference points, but join and rejoin those in a self-organized man-

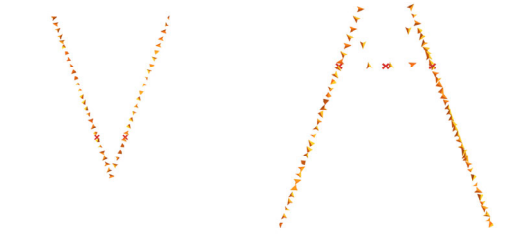


(a) Single Agent Set Approach ($\mathtt{C}^{*}_{\mathtt{ring}}$)

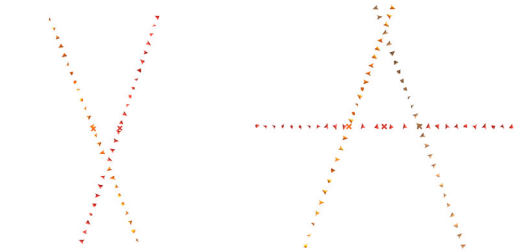


(b) Multi Agent Set Approach ($\mathtt{C}^{+}_{\mathtt{ring}}$)

Fig. 7 Olympic Rings Pattern experiment



(a) $\mathtt{C}^{*}_{\mathtt{line}}$ for letter V (b) $\mathtt{C}^{*}_{\mathtt{line}}$ for letter A



(c) $\mathtt{C}^{+}_{\mathtt{line}}$ for letter V (d) $\mathtt{C}^{+}_{\mathtt{line}}$ for letter A

Fig. 8 Letter Pattern experiment, aiming to create the Letters V and A as its visual effect on macro level

ner. Nevertheless, from a macro perspective, the described Olympic Ring formation becomes visible.

When switching to the Multi Agent Set Approach with $\mathtt{C}^{+}_{\mathtt{ring}}$, we can identify a qualitative difference to $\mathtt{C}^{*}_{\mathtt{ring}}$. Because $\mathtt{C}^{+}_{\mathtt{ring}}$ forces agents to permanently assign one reference point, agents now are distributed randomly but thus evenly on the five rings, stabilizing the visual effect (cf. Fig. 7b). For experiment parameters see Table 1.

5.2.3 The letter patterns

In the third experiment using multiple reference points, we use the `Line-of-Fliers` swarm behavior to create shapes resembling the letters *A* and *V*. This experiment demonstrates the varying effects that the Single Agent Set Approach can generate in comparison to the Multi Agent Set Approach. The letter *V* is formed through the alignment of two reference points, whose rotation is chosen such that their corresponding lines intersect at an apex point. To achieve symmetry, the two reference points and the apex point should form an isosceles triangle. Utilizing this reference point setup with the Single Agent Set Approach $\mathtt{C}^{*}_{\mathtt{line}}$, we can observe the swarm creating a pattern that closely resembles the letter V (cf. Fig. 8a). The transition to the Multi Agent Set Approach, denoted by $\mathtt{C}^{+}_{\mathtt{line}}$, results in a transformation of the pattern, wherein the lines now intersect to form a cross-like configuration (cf. Fig. 8c).

Forming the letter *A* requires extending the previously described alignment of two reference points that form the letter *V* by a third reference point. This third reference point is placed equidistant between the two other reference points and aligned such that its line intersects both of them. Employing the reference point setup with the Single Agent Set Approach, denoted by $\mathtt{C}^{*}_{\mathtt{line}}$, we see that the swarm is incapable of generating the complete arch and the horizontal bar of the letter in a sufficient way (cf. Fig. 8b). Compared to that, the effect on macro level produced by the Multi Agent Set Approach $\mathtt{C}^{+}_{\mathtt{line}}$ comes quite closer to forming a readable letter *A* (cf. Fig. 8d). For experiment parameters see Table 1.

5.2.4 Ring-of-Rings

In a fourth experiment using hierarchies, we demonstrate the visual effects of the hierarchic swarm concept (cf. Sect. 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 but still keeping required computational resources limited. To illustrate the effects, we again use the `Ring-of-Fliers` swarm behavior on all hierarchic layers and for all relevant reference points, i.e., $\mathtt{C}^{\mathtt{H}*}_{\mathtt{ring},\mathtt{ring},\mathtt{ring}}$. Like in the other experiments, switching from the Single Agent Set Approach to the Multi Agent Set Approach produces different effects on the macro level. As within the second experiment (cf. Sect. 5.2.2), the use of the Single Agent Set Approach can result in unbalanced rings (cf. Fig. 9a). While the ring to the left of the figure appears to be very crowded and agents on `layer 2` in that part form a 'filled-flower'-like structure around their transitive reference point on `layer 0`,

Table 2 Parameters of the Ring-of-Rings (RoR) and Ring-of-Line-of-Rings (RoLoR) experiment. # agent indicates the total amount of agents and the count of agents per reference point in parentheses

Hierarchic layer	# agents	com-radius	max-velocity	calc-function	$r_{\texttt{ring}}$
`RoR layer 0`	4 (4)	25	0.2	$C^{+}_{\texttt{ring}} \setminus C^{*}_{\texttt{ring}}$	20
`RoR layer 1`	16 (4)	20	0.2	$C^{+}_{\texttt{ring}} \setminus C^{*}_{\texttt{ring}}$	15
`RoR layer 2`	128 (8)	10	0.2	$C^{+}_{\texttt{ring}} \setminus C^{*}_{\texttt{ring}}$	10
`RoLoR layer 0`	8 (8)	20	0.3	$C^{+}_{\texttt{ring}} \setminus C^{*}_{\texttt{ring}}$	20
`RoLoR layer 1`	64 (8)	5	0.3	$C^{+}_{\texttt{line}} \setminus C^{*}_{\texttt{line}}$	10
`RoLoR layer 2`	384 (6)	5	0.3	$C^{+}_{\texttt{ring}} \setminus C^{*}_{\texttt{ring}}$	7

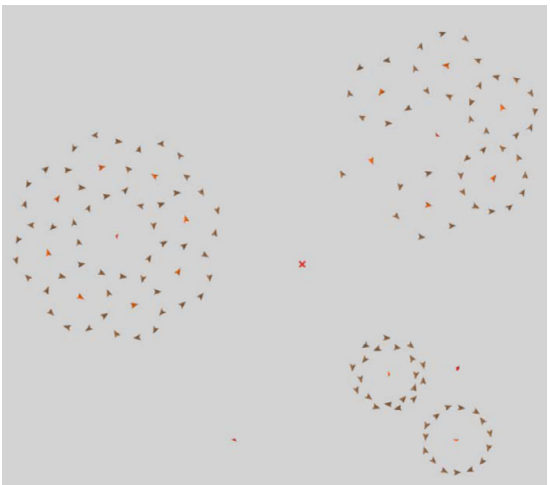
i.e., the agent of `layer 0` in the middle-left, other agents of `layer 0` stay isolated (cf. the agent placed at the bottom of the figure). The Multi Agent Set Approach creates balanced rings as expected (cf. Fig. 9b). On each layer of the swarm the exact number of commanded agents participate in the respective structure. For experiment parameters see Table 2.

5.2.5 Ring-of-Line-of-Rings

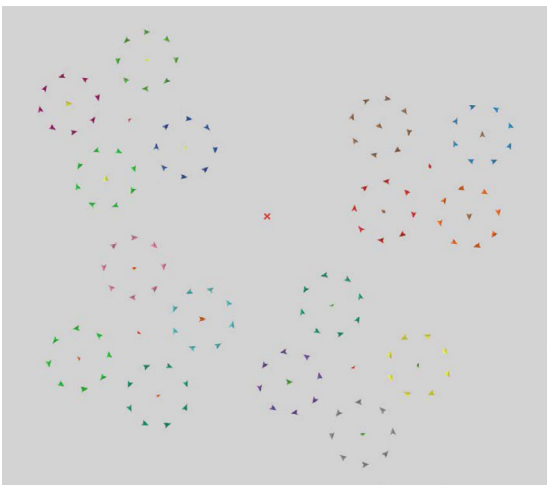
This experiment illustrates the effects of the hierarchical swarm concept with different swarm behaviors on different layers. The swarm in this experiment has three layers. A `Ring-of-Fliers` behavior is used on `layer 0` and `layer 2`, while a `Line-of-Fliers` is used on `layer 1`. This configuration yields towers of rings arranged in a ring-shaped configuration (cf. Fig. 10). Switching from the Multi Agent Set Approach to the Single Agent Set Approach results in similar effects as described in Sect. 5.2.2 and Sect. 5.2.4: The visual components we can observe on macro layer are unbalanced in terms of the number of agents forming the respective part of the overall shape which results in smaller towers and unbalanced rings. For experiment parameters see Table 2.

5.2.6 Adaptation to disturbances

In a further experiment, we demonstrate the adaptivity of our approach in a dynamic environment. Therefore, we allow for stressing the system with external disturbances for all different swarm behaviors we discussed before. To make this disturbance clearly visible and also allow for further studying the adaptivity of our approach, we integrate the possibility to place static obstacles in the environment which influence the agents' behavior. Agents are able to detect these obstacles in their neighborhood and avoid them while executing their swarm behavior by taking them into account within their separation urge. In Fig. 11, we can observe a multi-reference-point swarm executing the `Ring-of-Fliers` behavior each while avoiding an obstacle placed in the middle of the structure.



(a) Single agent set ($C^{*}_{\texttt{ring}}$)

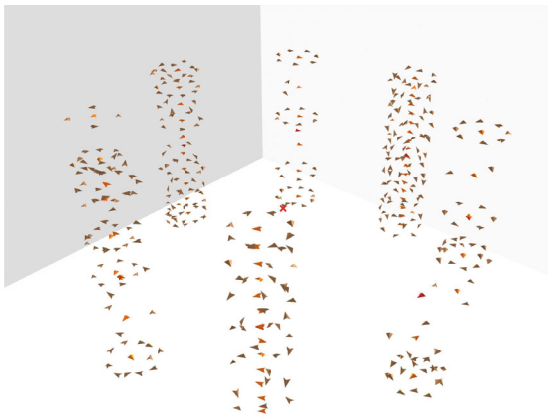


(b) Multi agent set ($C^{+}_{\texttt{ring}}$)

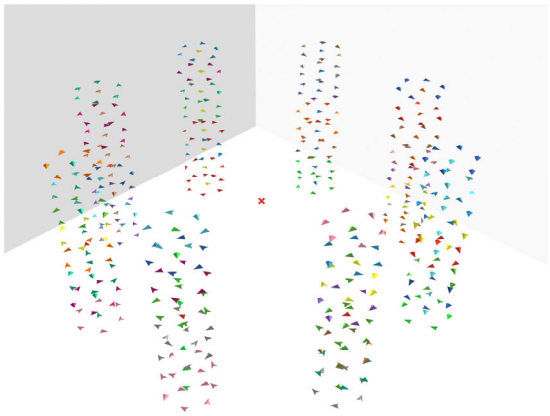
Fig. 9 Ring-of-Rings experiment producing a solar system-like visual effect using the hierarchic swarm concept

5.3 Empirical study on adaptivity

To empirically evaluate the adaptivity of PROTEASE 2.0, we conducted a series of experiments AE1—AE5, covering the



(a) Single Agent Set Approach $C^{H*}_{\texttt{ring,line,ring}}$



(b) Multi Agent Set Approach $C^{H+}_{\texttt{ring,line,ring}}$

Fig. 10 3D visualization of the Ring-of-Line-of-Rings experiment

base swarm behaviors (`Ring-of-Fliers` in `AE1`, `Line-of-Fliers` in `AE2`, `Ball-of-Fliers` in `AE3`), multiple reference points with Single Agent Set (Flower Pattern in `AE4`) and Multi Agent Set (Olympic Rings Pattern in `AE5`), and hierarchical swarm structures (ring of line of rings `AE6`). To give evidence of the adaptivity of our approach, we stress each of the different setups with system-internal and external disturbances (`D1`—`D5`). For a systematic analysis, we structured each experiment with the same series of events to stress the adaptability properties of our approach during an execution of 2500 simulation ticks.

- **Initial Stabilization (`D1`):** The swarm is allowed to stabilize and reach a steady state starting from a randomly initialized agent distribution. Thus, we can determine the base-line measures (see below) for the respective swarm's properties in a steady-state.
- **Scalability (`D2:`)** We integrate 20 new agents from random initial positions at tick 500.
- **Robustness (`D3`):** We remove a subset of 20 agents randomly from the swarm at tick 1000.

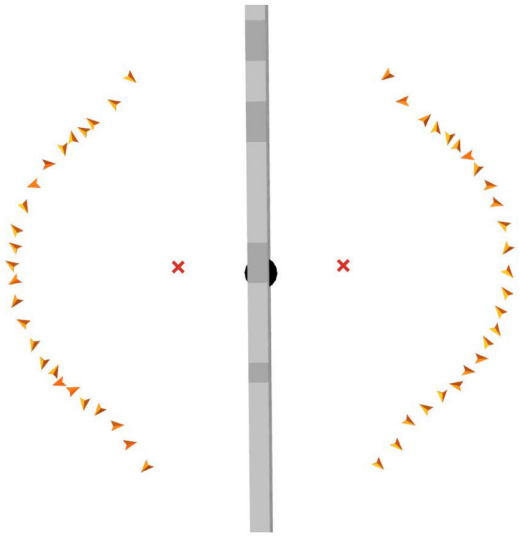


Fig. 11 Obstacle disturbance placed in the middle of a Multi Reference Point Flower ($C^{*}_{\texttt{ring}}$)

- **Flexibility (`D4`):** We move reference points to a new position at tick 1500.
- **Environment (`D5`):** We place a static obstacle in the environment at tick 2000.

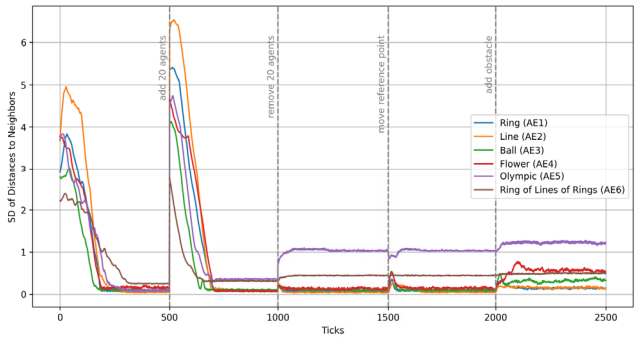
For quantifying the self-adaptation property of `PROTEASE` 2.0, we determine different measures: **(1)** the standard deviation of euclidean distances between closest agents (*Neigborhood-Stabilization*), **(2)** the standard deviation of agents' euclidean distances to their reference point (*Reference-Stabilization*), and **(3)** the velocity of the center of gravity (*Swarm-Movement*) at every execution round measured over a sliding window of 10 NetLogo simulation ticks. For all three measures **(1)—(3)**, we can deduce a stabilized system, i.e., one that successfully adapted to the disturbance, when the respective measure (m) stabilizes after the disturbance. We define stabilization in tick t_n with $n \in [1..2500]$, if the measure m does not change significantly within the next 50 ticks, i.e.,

$$\forall x \in [1..50] : \texttt{abs}(m(t_n) - m(t_{n+x})) \leq \delta \qquad (15)$$

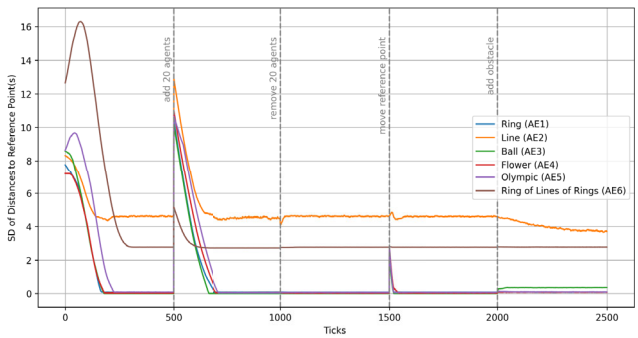
with `abs` beeing the absolute value and $\delta = 0.05$. For each experiment, we also recorded the amount of ticks the swarm requires to recover from the respective disturbance. We allow the reader to reproduce our experiments using our implementation provided on GitHub, where all experiments are preconfigured and thus can be easily re-executed and modified and interpreted.

5.3.1 Results

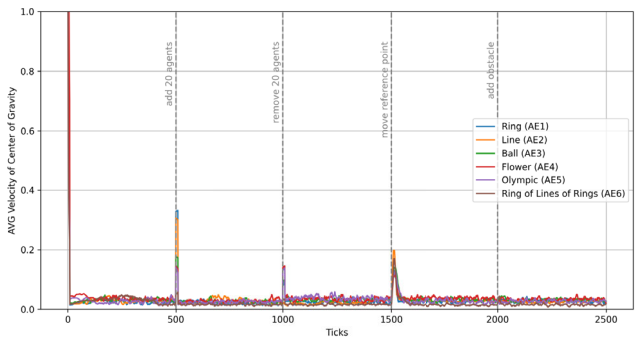
We present the results of our experiments in Figs. 12a to 12c, where we plot the different measures **(1)—(3)**, each for an



(a) SD of Distances to Neighbors



(b) SD of Distances to Reference Point(s)



(c) AVG Velocity of Center of Gravity

Fig. 12 Empirical evaluation of adaptivity. Measured for all agents in experiments AE1— AE6 (Standard Deviation as SD, Average as AVG). Scaling of y-axis is different for each figure

exemplary run and during the whole execution of the experiment. This enables us to observe the swarm's behavior after each disturbance D1—D5. We systematically provide stabilization times for AE1—AE6 and the values after the first stabilization for the different disturbances and measures in Table 3.

(1) Neighborhood-stabilization In Fig. 12a, we can observe very similar trends for each experiment AE1—AE6. After disturbances D1—D4 (not D5), each swarms takes a different amount of ticks to stabilize (cf. Table 3) but eventually reaches a steady state for our stability metric defined above. Despite this similarity in trends, we see different values for measure (1) in AE1— AE5 after the events D1— D4 with AE6 showing the most prominent changes (cf. Table 3). While the introduction of new agents initially in D1 or in D2 increases the scattering within all swarms (i.e., a maximum value > 6 for AE2 after D2) producing long stabilization times (306 ticks for AE6 after D1), the removal of agents in D3 and the movement of reference point(s) in D4 has low (i.e., a maximum value of 55 ticks for AE5 after D4) to no such effect (cf. AE6 after D4). Introducing obstacles in D5, instead, shows different effects. AE2, AE5, and AE6 stabilize quickly again (i.e., with a maximum value of 27 ticks for AE5) while AE1, AE3 and AE4 show a more prolonged disturbance effect (cf. 136 ticks for AE4).

(2) Reference-stabilization Overall, observable trends in AE1— AE6 are similar to that of *(1) Neighborhood-Stabilization*. Stabilization times after disturbances vary between most experiments but stay in a similar range. We identify an outlier with AE2, wich showcases a different general behavior (cf. Fig. 12b and Table 3). Especially after D2, AE2 does not recover from the disturbance while all other experiment settings achieve such stabilization. Also after D5, AE2 takes a long time for stabilization (245), while all others stabilize very quickly (AE5 shows the longest stabilization time of 29 ticks). Similar to *Neigborhood-Stabilization*, we see different stabilzation levels. While swarm behavior (complex or not), that is based only on "ring" or "ball" behavior (AE1, AE3, AE4, AE5) tends to stabilize close to 0 (cf. Table 3), such swarm behavior that also involves "line" behavior (AE2, AE6) stabilizes at higher values (e.g., around 2.75 for AE6 for D1—D5, cf. Table 3). Concerning the level of disturbance and the respective time taken to recover, we can observe that D1 and D2 have a high impact, disturbances D3 and D5 have low (i.e., 29 ticks in AE3, cf. Table 3) to no impact (i.e., AE1, AE3, AE4, AE5, cf. Table 3). Disturbance D4 seems to have a medium impact on AE1, AE3, AE4, AE5, and AE2 (maximum of 71 ticks for AE2) but close to no impact on AE6 (2 ticks).

(3) Swarm-movement Regarding the overall movement of the swarm's center of gravity, we can observe that all swarms stabilize very quickly after each disturbance with a maximum time of 34 ticks after D4 for AE3 (cf. Fig. 12c and Table 3). Overall, despite initialization in D1, all disturbances do not have a great impact on the overall movement of the swarm's center of gravity. Especially for D5, we can not observe any disturbance effect at all.

5.3.2 Interpretation

The results of our experiments reveal several insights into the stabilization behavior of PROTEASE 2.0 under different disturbances which we can analyze in accordance with terminology from system dynamics theory [45].

Table 3 Stabilization time in ticks T, values M for measure *Neighborhood-Stabilization **(1)**,* Reference-Stabilization *(2), and* Swarm-Movement *(3)* after disturbances D1—D5 for experiments AE1—AE6 as T/M

Exp.	D1	D2	D3	D4	D5
(1)					
AE 1	186/0.12	200/0.11	20/0.11	34/0.11	59/0.18
AE 2	247/0.09	190/0.12	19/0.08	17/0.24	8/0.16
AE 3	174/0.12	164/0.13	24/0.15	46/0.12	33/0.27
AE 4	188/0.16	210/0.10	14/0.13	49/0.18	136/0.58
AE 5	272/0.13	198/0.34	53/1.04	55/1.04	27/1.20
AE 6	306/0.23	148/0.35	8/0.39	0/0.44	2/0.44
(2)					
AE 1	166/0.11	187/0.12	0/0.08	19/0.12	0/0.07
AE 2	216/4.57	-/-	22/4.63	71/4.61	245/3.91
AE 3	177/9.95	160/0.03	0/0.00	20/0.01	29/0.31
AE 4	180/0.05	202/0.07	0/0.52	39/0.06	0/0.08
AE 5	223/0.11	214/0.09	0/0.50	19/0.10	0/0.12
AE 6	290/2.82	119/2.77	0/2.75	2/2.81	0/2.78
(3)					
AE 1	10/0.01	10/0.03	10/0.02	24/0.07	0/0.03
AE 2	10/0.02	10/0.02	0/0.02	22/0.08	0/0.03
AE 3	10/0.02	10/0.02	10/0.03	34/0.07	0/0.03
AE 4	10/0.04	10/0.03	10/0.03	5/0.07	0/0.03
AE 5	10/0.04	10/0.02	10/0.03	32/0.07	0/0.03
AE 6	10/0.02	0/0.05	0/0.03	23/0.07	0/0.02

(1) Neighborhood stabilization The stabilization times observed in experiments AE1—AE6 after disturbances D1—D4 are influenced by the varying number of agents in each experiment. Changes in the standard deviation (SD) values for stabilization can be attributed to the random nature of disturbances. For instance, removing different numbers of agents from various reference points leads to non-equal distribution among neighbors assigned to different reference points (a ring with radius r shows different distances between evenly distributed agents, if agent numbers vary). Initialization (D1) and the introdu ction of new agents (D2) have a significant impact on scattering due to the large distances between newly added agents and the rest of the swarm that is already in formation. In contrast, disturbances D3 and D4 exhibit only minimal impact, as the remaining agents in the swarm quickly adapt collectively as described in Sect. 4.4. The rapid stabilization observed in AE2, AE5, and AE6 can be explained by the fact that their formations are not interconnected (i.e., neighbors are not influenced at all), or, in the case of AE2, the obstacle directly only affects two neighbors (the effect on other agents is lower beacause their local neighborhood is influenced only on a minimal level). Conversely, AE1, AE3, and AE4 are more significantly influenced, as the obstacle disrupts a previously connected formation, impacting a larger number of agents.

(2) Reference-stabilization The stabilization times for the distance to reference points also vary across experiments due to differences in agent numbers (the higher the agent number, the higher the time consumed for stabilization). Peaks in SD values after D1 and D2 result from the random placement of new agents, which initially have large distances to their reference points. The relatively long and unequal stabilization times observed in some cases (cf. AE1—AE6 in D1 and D2) can be attributed to this random initialization and the varying agent numbers. The reason for the fact that formations involving "line" swarm behavior exhibit greater changes in SD values compared to "ring" and "ball" behaviors can be explained by the intended, non-uniform distances between agents and reference points within "line"-based formations (some agents are placed close to the reference point, others with greater distances along the line). The prolonged stabilization time for AE2 after D5 can be explained by the center urge u_c and separation urge u_s inherent working against each other in the Line-of-Fliers, which is further intensified by the obstacle's influence on flight direction (u_o).

(3) Swarm-movement The overall movement of the swarm's center of gravity exhibits relatively low levels of disturbance across all experiments. This can be attributed to several factors. During initialization (D1), the random placement of agents already has a stabilizing effect on the measure because a fair random generator for positions results in a centric center of gravity. Also, their immediate position adaptation towards their reference points result in quick stabilization as all agents in the swarm instantly head for the reference point's position, enforced by the center urge's influence dominating the others in the calculation functions (a positive feedback). A similar effect is observed after D2, where all new agents adapt rapidly while other agents do only adapt their positions marginally (negative feedback). Disturbance D3 has a comparable attenuating effect, as the removal of agents is quickly compensated for by the remaining swarm (negative feedback). In D4, the swarm adapts instantly to the new positions of the reference points as all center urges are influenced similarily for all agents and seperation urges are reduced as soon as the system as a whole starts movin (positive feedback, as soon as "front" agents free space for others when moving). Finally, D5 has no significant effect on the overall movement, as the obstacle is placed in the center of the formation and does not substantially influence the swarm's center of gravity (cf. Fig. 11).

5.4 Discussion

The proof-of-concepts experiments demonstrate the various use cases and the advantages and disadvantages of the Single Agent Set and Multi Agent Set approaches. Depending on the desired visual effect on macro level, each of the possible approaches of selecting the relevant reference point locally can be the one fitting better to that needs. For instance, the Single Agent Set Approach C^+ is better suited to create continuous shapes than the Multi Agent Set Approach, e.g., when using the `Line-of-Fliers` algorithm for the letter *V* (cf. Sect. 5.2.3) and the `Ring-of-Fliers` algorithm for the Flower Pattern (cf. Sect. 5.2.1). The disadvantage of this approach is that the distribution of the agents in not-connected forms is less controllable, which frequently results in imbalanced shapes, e.g., the imbalanced Olympic Rings (cf. Sect. 5.2.2) and the different sized towers in the Ring-of-Line-of-Rings experiment (cf. Sect. 5.2.5). In contrast, the Multi Agent Set Approach C^* is more appropriate for the creation of distinct intersections, e.g., when using the `Line-of-Fliers` algorithm for the letter *A* (cf. Sect. 5.2.3). It is also best used for the creation of multiple separate shapes, e.g., for creating balanced Olympic rings (cf. Sect. 5.2.2) or multiple towers of rings (cf. Sect. 5.2.5). The experiments also show how the hierarchic approach can create complex and visually interesting effects with both the Single Agent Set and Multi Agent Set Approaches (cf. Fig. 5a). Note that to capture the effects of this approach on static images, we used a single rotation for the algorithms on each layer, defined by the user-controlled reference point. Therefore we encourage the reader to use our implementation on GitHub to explore the dynamic effects without rotation synchronization. Our empirical evaluation of adaptivity (cf. Sect. 5.3) demonstrates that PROTEASE 2.0 effectively adapts to both internal and external disturbances across all tested scenarios. Agents were able to reconfigure themselves and maintain the desired swarm behavior, showcasing the robustness and flexibility of our approach. These findings validate the adaptivity of PROTEASE 2.0 and its suitability for dynamic and unpredictable environments.

While we have successfully implemented fundamental swarm behaviors for real drones – such as single reference point control and flat-layered swarm formations[4] – our objective is to extend the concepts presented in this paper to real-world drone applications. Bridging the gap between simulation and reality necessitates imposing certain constraints on swarm behavior, as well as on the resulting individual and collective movements, to mitigate physical limitations that typically hinder the unrestricted execution of swarm dynamics in real-world environments. Unlike in simulated swarm environments, where rotor-induced downwash effects are negligible, real-world deployments must account for such aerodynamic interactions. Although appropriate collision hulls in our collision avoidance mechanisms can help address potential issues during real-world experiments, these measures may also impose limitations on the visual effects observed in simulated experiments (cf. Sect. 5.2). To precisely replicate simulated swarm behaviors, a significantly larger operational space would be required to facilitate 'one-over-the-other' flight formations. In more constrained environments, such as our designated flight arena [46], it may be necessary to limit the degrees of freedom available to users when positioning reference points or adjusting the parameters of PROTEASE 2.0.

To come by this issue, we have already developed a mixed version of simulated and real drones that can be combined in swarms for flying in our arena. That way, we can simulate parts of the swarm, e.g., all agents despite those on the highest layer, so that the desired macro level effect becomes visible in the real world while unnecessary parts are only displayed in simulation, e.g., the Olympic Rings establish with real drones but their respective middle points stay simulated. Because flight-time limitations and frequent drone exchanges in normally stable formations can obscure the desired macro-level effect, we also initiated hardware development to increase drone uptime and improve battery charging strategies, e.g., by improving traffic management in drone loading areas or increasing the number of battery charging zones.

One measure, we did not take into account in our evaluations in Sect. 5.3 is that of disturbances concerning position measurements. While failures concerning position measurements for single drones frequently occur when using real-hardware, we found that this kind of disturbance does not affect the collective effect but only harms the individual drone where the failure occurs: If the failures are not permanent, the position controller and the Kalman filtering onboard the flight controller compensate for the failure. Otherwise, the effect is that the affected drone accelerates uncontrollably in a random direction and crashes, while the other drones adapt their neighborhood and compensate for the dropout in a self-organized manner (cf. agent removal in Sect. 4.4).

6 Conclusion

In this paper, we revisit our swarm concept, PROTEASE 2.0, with the aim of extending the range of decentralized formation flight patterns for drones, first introduced in [2]. Initially, we refine our understanding of parametrizable swarm behavior, where different emergent effects can arise from a single generalized implementation by merely altering the

[4] https://youtu.be/YTd9pCpeYuU?si=6iz-Wq5ibbsAjj8s.

parameters utilized by each swarm member. We then formalize basic swarm behaviors that realize ring, line, and ball formations around a reference point in a 3-dimensional environment. We demonstrate how PROTEASE 2.0 can generate new formations in three distinct ways: (1) by increasing the number of parallel swarms that interact, (2) by expanding the single reference point into a set of reference points, and (3) by introducing hierarchical swarms that integrate basic swarm behaviors with (1) and (2), thereby producing more complex phenomena observable at the macro level. While the primary focus of this paper is the experimental evaluation of these new swarm concepts, which enable novel flight patterns for drones in simulation, we also empirically evaluate on the adaptivity features of our approach giving proof for its self-stabilization properties and present initial proof-of-concept experiments using real drones to form simpler swarm patterns based on our concepts.

In future work, we will address the physical challenges associated with using real drones, such as battery constraints, rotor downwash, and collision avoidance. While we have already discussed these issues separately in previous research [3], our future efforts will integrate these challenges with PROTEASE 2.0. There, we aim for evaluating the same measures for self-stabilization and adaptivity as presented in Sect. 5.3 but now using real drones. Additionally, future work could shift the focus from formation flight to enhancing the discovery capabilities of our swarm behavior, for example, by applying hierarchical Particle Swarm Optimization to search for specific parameters. In this case, each particle would form its own swarm, aggregating and communicating local results to better survey larger regions. In addition, we plan to overcome the "bottom-up" paradigm of inventing new basic swarm behavior or combinations of them and rather aim for a "top-down" approach, where we define the desired swarm behavior on the macro level and then let the swarm members figure out how to achieve this behavior by themselves, which we think is possible by using AI techniques like reinforcement learning or genetic algorithms [47].

Funding information Open Access funding enabled and organized by Projekt DEAL.

References

1. 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 (2021)
2. Kosak, O., Kastenmüller, P., Wanninger, C., Reif, W.: An approach for extended swarm formation flight with drones: PROTEASE 2.0. In: Margaria, T., Steffen, B. (eds.) Leveraging Applications of Formal Methods, Verification and Validation. Rigorous Engineering of Collective Adaptive Systems, pp. 263–280. Springer, Cham (2025)
3. Kosak, O.: Mission programming for flying ensembles: combining planning with self-organization. Doctoralthesis, Universität Augsburg (2021)
4. Hoorn, R.A.: Plant proteases: from phenotypes to molecular mechanisms. Annu. Rev. Plant Biol. **59**, 191–223 (2008)
5. 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)
6. Zhang, Y., Wang, S., Ji, G.: A comprehensive survey on particle swarm optimization algorithm and its applications. Math. Probl. Eng. **2015** (2015)
7. Ma, M., Yang, Y.: Adaptive triangular deployment algorithm for unattended mobile sensor networks. IEEE Trans. Comput. **56**(7), 946–958 (2007)
8. 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 (2009)
9. 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 (2020)
10. Braubach, L., Pokahr, A.: Developing distributed systems with active components and jadex. Scalable Comp. Pract. Exp. 100–120 (2012)
11. 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)
12. 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. 2149–21543 (2004). https://doi.org/10.1109/IROS.2004.1389727
13. 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
14. 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. Aerosp. Sci. Technol. **98**, 105671 (2020)
15. Carbone, C., Garibaldi, O., Kurt, Z.: Swarm robotics as a solution to crops inspection for precision agriculture. KnE Eng. **3**(2), 552–562 (2018). https://doi.org/10.18502/keg.v3i1.1459
16. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. Comput. Graph. **21**(4), 25–34 (1987)
17. Rubenstein, M., Cornejo, A., Nagpal, R.: Programmable self-assembly in a thousand-robot swarm. Science **345**(6198), 795–799 (2014)
18. 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 (2020)
19. 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, Berlin (2002)
20. 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 (2018)
21. Sánchez-García, J., Reina, D.G., Toral, S.L.: A distributed pso-based exploration algorithm for a uav network assisting a disaster scenario. Future Gener. Comput. Syst. **90**, 129–148 (2019). https://doi.org/10.1016/j.future.2018.07.048
22. 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
23. 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, Cham (2020)
24. Lee, K.-B., Kim, Y.-J., Hong, Y.-D.: Real-time swarm search method for real-world quadcopter drones. Appl. Sci. **8**(7), 1169 (2018)
25. Trianni, V.: Coordinated motion. In: Evolutionary Swarm Robotics, pp. 73–95. Springer, Berlin (2008). https://doi.org/10.1007/978-3-540-77612-3_6
26. Pérez, I.F., Boumaza, A., Charpillet, F.: Learning collaborative foraging in a swarm of robots using embodied evolution. In: Artificial Life Conference Proceedings, vol. 14, pp. 161–162. MIT Press, Cambridge (2017)
27. 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., 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 Robot. Autom. Mag. **20**(4), 60–71 (2013). https://doi.org/10.1109/MRA.2013.2252996
28. Pianini, D., Viroli, M., Beal, J.: Protelis: practical aggregate programming. In: Proc. of the 30th Annual ACM Symposium on Applied Computing. SAC'15, pp. 1846–1853. ACM, New York (2015). https://doi.org/10.1145/2695664.2695913
29. 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, Proceedings, DisCoTec 2023, Lisbon, Portugal, June 19–23, 2023, pp. 31–51. Springer, Berlin (2023). https://doi.org/10.1007/978-3-031-35361-1_2
30. 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 (2016). https://doi.org/10.1109/IROS.2016.7759558
31. 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/IROS.2007.4399480
32. Varughese, J.C., Hornischer, H., Zahadat, P., Thenius, R., Wotawa, F., Schmickl, T.: A swarm design paradigm unifying swarm behaviors using minimalistic communication. Bioinspir. Biomim. **15**(3), 036005 (2020)
33. Bettini, L., Bourr, K., Pugliese, R., Tiezzi, F.: Coordinating and programming multiple ros-based robots with x-klaim. Int. J. Softw. Tools Technol. Transf. **25**(5), 747–764 (2023)
34. Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W.: Robot operating system 2: design, architecture, and uses in the wild. Sci. Robot. **7**(66), 6074 (2022). https://doi.org/10.1126/scirobotics.abm6074
35. Desai, A., Saha, I., Yang, J., Qadeer, S., Seshia, S.A.: Drona: a framework for safe distributed mobile robotics. In: 2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS), pp. 239–248 (2017)
36. Ghosh, R., Hsieh, C., Misailovic, S., Mitra, S.: Koord: a language for programming and verifying distributed robotics application. Proc. ACM Program. Lang. **4**(OOPSLA) (2020). https://doi.org/10.1145/3428300
37. Wolf, B., Chwala, C., Fersch, B., Garvelmann, J., Junkermann, W., Zeeman, M.J., Angerer, A., Adler, B., Beck, C., Brosy, C., Brugger, P., Emeis, S., Dannenmann, M., Roo, F.D., Diaz-Pines, E., Haas, E., Hagen, M., Hajnsek, I., Jacobeit, J., Jagdhuber, T., Kalthoff, N., Kiese, R., Kunstmann, H., Kosak, O., Krieg, R., Malchow, C., Mauder, M., Merz, R., Notarnicola, C., Philipp, A., Reif, W., Reineke, S., Rödiger, T., Ruehr, N., Schäfer, K., Schrön, M., Senatore, A., Shupe, H., Völksch, I., Wanninger, C., Zacharias, S., Schmid, H.P.: The scalex campaign: scale-crossing land surface and boundary layer processes in the tereno-prealpine observatory. Bull. Am. Meteorol. Soc. **98**(6), 1217–1234 (2017). https://doi.org/10.1175/BAMS-D-15-00277.1
38. Kosak, O., Wanninger, C., Angerer, A., Hoffmann, A., Schierl, A., Seebach, H.: Decentralized coordination of heterogeneous ensembles using jadex. In: IEEE 1st Int. Workshops on Found. and Appl. of Self* Systems (FAS*W), pp. 271–272 (2016). https://doi.org/10.1109/FAS-W.2016.65
39. Kosak, O., Wanninger, C., Angerer, A., Hoffmann, A., Schiendorfer, A., Seebach, H.: Towards self-organizing swarms of reconfigurable self-aware robots. In: Found. and Applications of Self* Systems, IEEE Int. Workshops on, pp. 204–209. IEEE Press, New York (2016)
40. Schörner, M., Wanninger, C., Hoffmann, A., Kosak, O., Ponsar, H., Reif, W.: Modeling and execution of coordinated missions in reconfigurable robot ensembles. In: 2020 Fourth IEEE International Conference on Robotic Computing (IRC), pp. 290–293 (2020). https://doi.org/10.1109/IRC.2020.00053
41. Kosak, O., Bohn, F., Keller, F., Ponsar, H., Reif, W.: Ensemble programming for multipotent systems. In: 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W) (2019)
42. Wanninger, C., Alfano, L., Schörner, M., Hoffmann, A., Kosak, O., Reif, W.: under review - Semantic Plug and Play: an Architecture Combining Linked Data and Reconfigurable Hardware. In: 16th IEEE International Conference on Semantic Computing. (ICSC). IEEE Press (2019)
43. Schörner, M., Wanninger, C., Hoffmann, A., Kosak, O., Reif, W.: Architecture for emergency control of autonomous uav ensembles. In: 2021 IEEE/ACM 3rd International Workshop on Robotics Software Engineering (RoSE), pp. 41–46 (2021). https://doi.org/10.1109/RoSE52553.2021.00014
44. 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 (2020)
45. Pruyt, E.: Small System Dynamics Models for Big Issues: Triple Jump Towards Real-World Complexity. TU Delft Library, Delft (2013)

46. 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
47. Alharbi, A.M., Alshehri, G., Elhag, S.: Reinforcement learning of emerging swarm technologies: a literature review. In: Arai, K. (ed.) Proceedings of the Future Technologies Conference (FTC) 2024, vol. 3, pp. 478–494. Springer, Cham (2024)