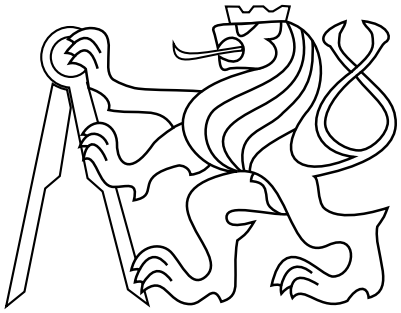
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Flocking of Unmanned Aerial Vehicles Based on Semantic Visual Perception

Bachelor's Thesis

Tomáš David

Prague, May 2025

Study programme: Open Informatics
Specialisation: Artificial Intelligence and Computer Science

Supervisor: Ing. Martin Jiroušek

Acknowledgments

Firstly, I would like to thank my family for their continued support during my studies. I would also like to express my gratitude to my supervisor Ing. Martin Jiroušek for his guidance and feedback throughout the creation of this thesis. Furthermore, my thanks goes to the Multi-robot Systems group and its members for their support during the real-world experiments.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **David Tomáš** Personal ID number: **516523**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence and Computer Science**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Flocking of Unmanned Aerial Vehicles Based on Semantic Visual Perception

Bachelor’s thesis title in Czech:

Hejno bezpilotních letounů založené na sémantickém vizuálním vnímání

Guidelines:

The thesis addresses the problem of decentralized control of a swarm of non-communicating drones (multirotor unmanned aerial vehicles). The objective is to extend the algorithm described in [1], which was originally designed for ground robots, to a three-dimensional environment. The algorithm utilizes a semantic visual map generated by a sensor system and subsequent data processing. The output of the algorithm corresponds to the required acceleration of each drone. The resulting control algorithm will be implemented in C/C++ as part of the control module of the MRS UAV system described in [3].
Student tasks:
1) Familiarize yourself with the theoretical and practical aspects of multi-robot swarm systems.
2) Study the article [1] in detail and propose an appropriate approach for extending the original 2D algorithm to a 3D environment.
3) Implement the proposed solution in C/C++ and integrate it as a control module within the MRS UAV system.
4) Verify the functionality of the algorithm in an appropriate simulation environment.
5) If the hardware allows, conduct an experimental validation of the algorithm on real platforms.

Bibliography / sources:

[1] Mezey, D., Bastien, R., Zheng, Y., McKee, N., Stoll, D., Hamann, H., & Romanczuk, P. (2024). Purely vision-based collective movement of robots. arXiv. https://arxiv.org/abs/2406.17106
[2] S. -J. Chung, A. A. Paranjape, P. Dames, S. Shen and V. Kumar, "A Survey on Aerial Swarm Robotics," in IEEE Transactions on Robotics, vol. 34, no. 4, pp. 837-855, Aug. 2018, doi: 10.1109/TRO.2018.2857475
[3] Baca, T., Petrlik, M., Vrba, M. et al. The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles. J Intell Robot Syst 102, 26 (2021). https://doi.org/10.1007/s10846-021-01383-5
[4] V. Walter, N. Staub, A. Franchi and M. Saska, "UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs," in IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 2637-2644, July 2019, doi: 10.1109/LRA.2019.2901683

Name and workplace of bachelor’s thesis supervisor:

Ing. Martin Jiroušek Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **23.01.2025** Deadline for bachelor thesis submission: _____

Assignment valid until: **20.09.2026**

prof. Dr. Ing. Jan Kybic
Head of department’s signature

prof. Mgr. Petr Páta, Ph.D.
Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

FAKULTA ELEKTROTECHNICKÁ
FACULTY OF ELECTRICAL ENGINEERING
Technická 2
166 27 Praha 6

DECLARATION

I, the undersigned

Student's surname, given name(s): David Tomáš
Personal number: 516523
Programme name: Open Informatics

declare that I have elaborated the bachelor's thesis entitled

Flocking of Unmanned Aerial Vehicles Based on Semantic Visual Perception

independently, and have cited all information sources used in accordance with the Methodological Instruction on the Observance of Ethical Principles in the Preparation of University Theses and with the Framework Rules for the Use of Artificial Intelligence at CTU for Academic and Pedagogical Purposes in Bachelor's and Continuing Master's Programmes.

I declare that I used artificial intelligence tools during the preparation and writing of this thesis. I verified the generated content. I hereby confirm that I am aware of the fact that I am fully responsible for the contents of the thesis.

In Prague on 19.05.2025

Tomáš David
.....
student's signature

Abstract

The aim of this work is to design, implement and evaluate a distributed Unmanned Aerial Vehicle (UAV) swarm controller that operates on semantic visual maps of the agent's surroundings. Traditional approaches to robot swarming often require knowledge of the positions of neighboring agents, which can be difficult to obtain reliably. In contrast, the proposed controller operates directly on the obtained semantic maps, thus completely removing the need for relative localization. The proposed controller was implemented in the C++ programming language, and thoroughly evaluated in simulators across a variety of scenarios and use cases. These simulations prove the controller is able to maintain swarm cohesion, align movement of agents, and prevent collisions. Additionally, the simulated results were also verified with real-world UAV swarm experiments.

Keywords Unmanned Aerial Vehicles, Swarm Robotics, Distributed control, Semantic Perception, Bio-inspired Robotics

Abstrakt

Cílem této práce je návrh, implementace a vyhodnocení distribuovaného řídicího systému pro roje bezpilotních prostředků (UAV), založeného na sémantických vizuálních mapách okolí jednotlivých agentů. Tradiční přístupy k robotickým rojům často vyžadují informace o polohách sousedních agentů, avšak získávat tato data spolehlivě může být obtížné. Na rozdíl od těchto přístupů využívá navržený řídicí systém k řízení přímo vstupní sémantické mapy a odstraňuje tak potřebu relativní lokalizace. Řídicí systém byl implementován v programovacím jazyce C++ a ověřen v různých simulovaných scénářích. Tyto simulace ukazují, že navržený řídicí systém je schopen udržovat soudržnost roje, zarovnat pohyb agentů a zabránit kolizím. Simulované výsledky byly navíc ověřeny v reálných experimentech s rojem UAV.

Klíčová slova Bezpilotní prostředky, rojová robotika, distribuované řízení, sémantické vnímání, bio-inspirovaná robotika

Abbreviations

FADAR Fake Detection and Ranging (see Section 3.2.2)

FOV Field of View

GPS Global Positioning System

LiDAR Light Detection and Ranging

MRS Multi-robot Systems Group

PID Proportional–Integral–Derivative (controller)

ROS Robot Operating System

RTK Real-time Kinematics

UAV Unmanned Aerial Vehicle

UVDAR UltraViolet Direction And Ranging

Contents

1 Introduction **1**
1.1 Problem statement . . . 1
1.2 Mathematical notation . . . 2
1.3 Reference frames . . . 2

2 Swarm controller **4**
2.1 Observations . . . 4
2.2 Control outputs . . . 5
2.3 Visual controller . . . 6
2.4 Augmented controller . . . 8
2.4.1 Drag controller . . . 8
2.4.2 Goal controllers . . . 9
2.4.3 Self-propelled movement . . . 9
2.4.4 Limited turning rate . . . 10
2.4.5 Wall repulsion . . . 10

3 Implementation **12**
3.1 Implementation of the swarm controller . . . 12
3.1.1 Simulating observations . . . 12
3.1.2 Blind spots . . . 12
3.1.3 Mitigation of blind spots . . . 13
3.2 Real-world setup . . . 13
3.2.1 Drift mitigation . . . 13
3.2.2 FADAR . . . 13

4 Evaluation methodology **15**
4.1 Custom swarm simulator . . . 15
4.2 MRS Multirotor Simulator . . . 15
4.3 Metrics . . . 15
4.3.1 Distances . . . 16
4.3.2 Polarization . . . 16
4.3.3 Swarm velocity . . . 17

5 Results **18**
5.1 Parametrization . . . 18
5.2 Simulated scenarios – custom simulator . . . 18
5.2.1 No goal . . . 18
5.2.2 Shared goal . . . 19
5.2.3 Leader(s)–followers . . . 21

5.2.4 Emergent behavior . . . 23
5.3 Robustness . . . 24
5.4 Limited FOV observations . . . 26
5.4.1 Vertical blind spots . . . 26
5.4.2 UVDAR blind spots . . . 26
5.5 MRS Multirotor Simulator . . . 26
5.5.1 Limited range of detection . . . 27
5.5.2 Results of tested scenarios . . . 28
5.6 Real-world experiments . . . 28
5.6.1 Description of experiments . . . 28
5.6.2 Shared goal . . . 29
5.6.3 Leader–follower . . . 30

6 Conclusion **33**
6.1 Future work . . . 33

7 References **34**

A Parameters **36**

B Supplementary materials **37**

C Use of AI **38**

1 Introduction

Swarms of Unmanned Aerial Vehicles (UAVs), and robots more generally, are an active topic of research because of their potential application in countless different scenario, including search and rescue [1], mapping [2], monitoring [3] and many others. Compared to a single UAV, swarms bring many advantages, such as reliability, ability to divide objectives and potential to carry more specialized equipment with heterogenous agents in the swarm. To be able to complete their goals safely and efficiently, individual agents need to coordinate their movement with other members of the swarm, otherwise, the swarm could tear apart or the agents could collide.

One of the approaches to coordinate the individual agents are algorithms based on the boids model introduced by Reynolds [4], which was created for simulating swarms of animals for use in computer graphics. This model represents agents as particles and includes rules for collision avoidance, velocity matching and flock centering, which together create a behavior similar to a flock of birds. Similar model was introduced by Vicsek et al. [5] and was used to study the emergent alignment of motion. Once again, the agents are represented as particles, however, in this model, the particles have a fixed speed and the model contains only a single rule for aligning the heading of the particles.

There are many approaches based on the Reynolds's boids model that were analyzed for their theoretical properties such as stability under different conditions [6], [7]. Similar approaches were also evaluated in real world experiments, example of which is the adaptation by Saska et al. [8] in which the boids algorithm was combined with a visual relative localization system to stabilize a swarm in an environment with obstacles.

Another way of modeling the interactions for coordinating swarms of robots are approaches based on Lloyd's algorithm [9], which use the geometric properties of Voronoi cells to guarantee collision avoidance. Uses of this approach include simulations of multi-target search and tracking by Dames [10] and space coverage and flocking by Boldrer et al. [11].

One potential drawback of all the approaches mentioned above is that agents require position and/or velocity estimates of the other agents to calculate their actions. This means that robots using these models must use systems to either communicate the data or estimate it using onboard systems.

A different model, that does not have this limitation, was proposed by Bastien & Romanczuk [12]. In their model the behavior of the swarm is controlled with visual semantic maps as inputs, completely removing the need for position estimation of other agents in the swarm. Their work is primarily concerned with agents moving in 2D environments and mentions extensions to 3D only briefly. This purely vision based model was further explored by Mezey et al. [13], who tested the impact of blind spots in the observations and performed real-world experiments with ground robots.

1.1 Problem statement

The aim of this work is to adapt the purely vision based swarming model proposed by Bastien & Romanczuk [12] for use in multirotor UAV swarming and evaluate its performance. This was achieved by solving the following subproblems:

(i) **Extension of the controller suitable for the control of UAVs**. This includes updating the observations to work well in a 3D environment and changing the control outputs to better match the capabilities of the UAVs. Furthermore, a method for including external control is proposed to allow for completing specified tasks.

(ii) **Implementation of the proposed controller.** This consists of implementing the controller in the C++ programming language and integrating it into the MRS UAV System [14] to allow interactions with simulators as well as real UAVs. Furthermore, since designing a detection system capable of creating the required observations is beyond the scope of this thesis, the implementation has to also contain a simulation of such system.

(iii) **Evaluation of the controller in simulations.** This includes evaluation in a range of simulated scenarios to assess the performance of the controller in different conditions. The evaluation is done both by checking whether the controller is able to stabilize the swarm and perform a given task as well as by quantifying the behavior to compare the results of individual experiments.

(iv) **Real-world experiments with the controller.** This requires preparing the controller for interactions with real UAVs, and evaluating the controller in real-world experiments to prove the validity of the simulated results.

1.2 Mathematical notation

Table 1.1 describes the mathematical notation used in this thesis.

Symbol	Meaning
lowercase or uppercase letter, e.g., a, X	scalar
lowercase bold letter, e.g., $\mathbf{x}$	column vector
uppercase bold letter, e.g., $\mathbf{X}$	matrix
Greek letter, e.g., θ, φ	angle
$\mathbf{x}^{\mathrm{T}}$, $\mathbf{X}^{\mathrm{T}}$	transposition of vector, matrix
r, c	row and column indexes in an image (range 0 to resolution $-$ 1)
$\mathbf{X}_{r,c}$	element at the row r and column c in the matrix $\mathbf{X}$ (row and column indexes are zero-based)
$\|\mathbf{x}\|$	Euclidean (L2) norm of the vector $\mathbf{x}$

Table 1.1: Mathematical notation.

1.3 Reference frames

This section defines all reference frames required for describing the the controller, its inputs and outputs. All of these reference frames are also visualized in Figure 1.1. First of the defined frames is the world frame $\mathcal{W}$, which has a fixed origin, x and y axes are horizontal and z is pointing up, creating a right handed coordinate system.

Each agent has a position defined as a vector in the world frame describing the center of the agent. The heading ψ of the agent is defined as the oriented angle between the positive x axis of the world frame $\mathcal{W}$ and the projection of the UAVs body frame[1] x axis into the world xy plane. For each agent, we also define a local cartesian frame $\mathcal{L}$ which is obtained

[1]The UAV body reference frame is a right handed coordinate system with origin in the center of the UAV and axes tightly connected to the physical frame of the UAV. The z axis points in the direction of thrust.

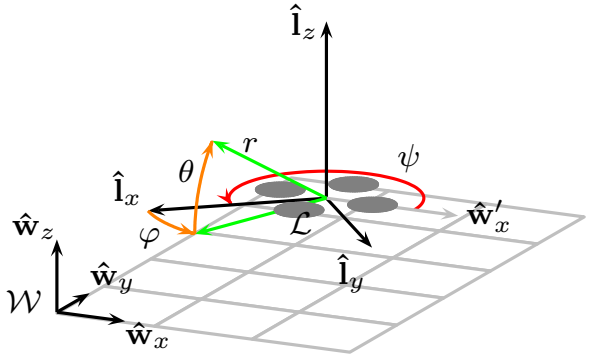


Figure 1.1: Reference frames used in this work. The $\hat{\mathbf{w}}_{x,y,z}$ and $\hat{\mathbf{l}}_{x,y,z}$ vectors point in the direction of the respective axes of the coordinate frames, but are not to scale. $\hat{\mathbf{w}}'_x$ has the direction of $\hat{\mathbf{w}}_x$ and is translated into the origin of frame $\mathcal{L}$.

by moving the origin of the coordinate system $\mathcal{W}$ into the center of the agent and rotating it around the z axis by ψ. The transform from $\mathcal{W}$ to $\mathcal{L}$ is

$$\begin{aligned} \mathbf{t}_{\mathcal{W}\to\mathcal{L}}(\mathbf{x}) &= \mathbf{R}_z(-\psi)(\mathbf{x} - \mathbf{p}_{\text{agent}}), \\ \mathbf{R}_z(\alpha) &= \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned} \tag{1.1}$$

Furthermore, for representing input observations, it is advantageous to define a local spherical frame $\mathcal{S}$ with coordinates $[r, \theta, \varphi]^{\mathrm{T}}$ and transformation to $\mathcal{L}$:

$$\mathbf{t}_{\mathcal{S}\to\mathcal{L}}\left(\begin{bmatrix} r \\ \theta \\ \varphi \end{bmatrix}\right) = \begin{bmatrix} r\cos(\theta)\cos(\varphi) \\ r\cos(\theta)\sin(\varphi) \\ r\sin(\theta) \end{bmatrix}. \tag{1.2}$$

In this work, the θ and φ angles are always used according to this definition of spherical coordinates.

2 Swarm controller

2.1 Observations

Each observation consists of projections of all[1] other agents onto a sphere around the observing agent. By doing this projection, we effectively remove the information about the distance of the observed agents and only keep the angles, where it is visible. The removal of distance is practical for some detection systems (e.g., cameras) that are not directly able to observe the distance, but can observe the angles. This projection can be interpreted as a function:

$$V(\theta,\varphi)=\begin{cases}1 & \text{some part of another agent is in the direction of } (\theta,\varphi) \text{ from}\\ & \text{the observing agent,}\\ 0 & \text{otherwise.}\end{cases} \tag{2.1}$$

For application in the controller, it is more practical to discretize it and transform it into a rectangular image (matrix). This allows using efficient data structures and algorithms for operations done by the controller. To create the observation image, we define the following mapping, which is used to get the spherical coordinate angles corresponding to a specific pixel in the image:

$$\theta(r)=\frac{-\pi r}{h-1}+\frac{\pi}{2}, \tag{2.2a}$$

$$\varphi(c)=2\pi\frac{c}{w}, \tag{2.2b}$$

where

r, c are row and column in the image,
h is the height (vertical resolution) of the image,
w is the width (horizontal resolution) of the image.

Combining the previous equations, the resulting observation image is a binary matrix

$$\mathbf{V}=[V(\theta(r),\varphi(c))]_{r\in(0,\dots,h-1),\,c\in(0,\dots,w-1)}. \tag{2.3}$$

An example observation is shown in Figure 2.1. Different resolutions of the observation image were tested and a resolution of 181 × 360 (height × width) points (1 pixel per degree) was chosen as a good compromise between quality and computation time.

In this work, all agents are represented as a sphere similarly to the original work by Bastien & Romanczuk [12]. The spherical shape of the agents was chosen because the visible shape and consequently the projected silhouette in the observation is not dependant on the direction, from which the sphere is observed. This also means that when two agents are moving relative to each other and stay same distance apart (e.g., one is orbiting the other), only the direction from which the agents is observed changes, but the silhouette stays the same. This property is advantageous for the stability of the swarm, as there are less sudden changes in the observed area and consequently less changes in the output. As an example of a shape that does not have this property, a cube shaped agent was tested. Figure 2.2 shows the relation of

[1] Some tests were performed with limited observation capabilities (e.g., limited range or Field of View (FOV)). In these cases, only projections of the visible agents are in the observation.

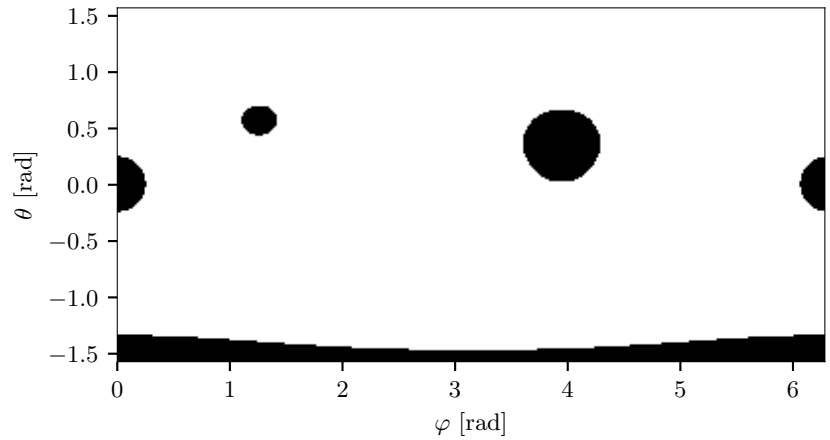
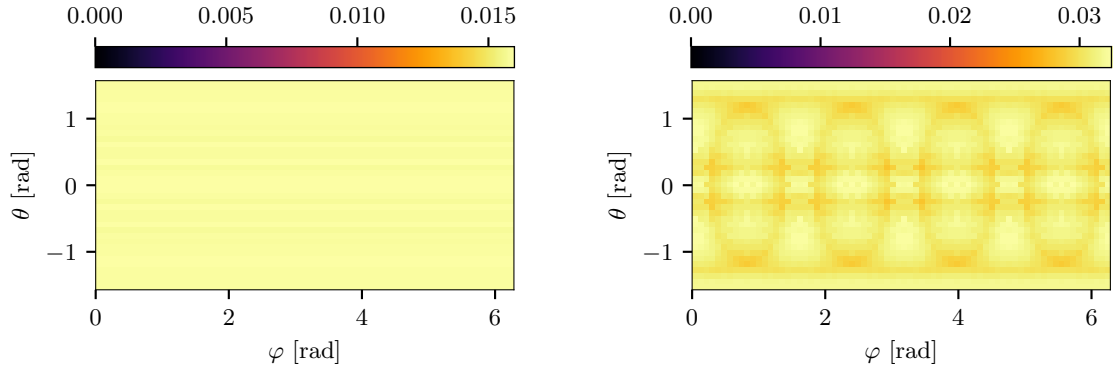


Figure 2.1: An example observation with four agents at observer's local coordinates: $[2, 0, 0]^{\mathrm{T}}$, $[1, 3, 2]^{\mathrm{T}}$, $[-1, -1, 0.5]^{\mathrm{T}}$, $[0.2, 0, -3]^{\mathrm{T}}$. Resolution: 181×360 pixels.



(a) Agent shape: sphere.

(b) Agent shape: cube.

Figure 2.2: Area of agent's projection observed at a fixed distance, from different direction. Angles are as defined in local spherical frame $\mathcal{S}$ of the observer, the observed agent has a fixed pose. Area unit is fraction of the observation area covered by the observed agent after factoring in the distortion caused by projection to the image.

the observed area to the angle from which the agent is observed. In the case of the spherical agents (Figure 2.2a), there are only minimal changes in the observed area, that are caused by the discretization of the observations[2]. As for the cube-shaped agents (Figure 2.2b), the changes in area are much more noticeable and as a result, asymmetrical responses are created, which may lower the stability of the swarm.

2.2 Control outputs

In the 2D model by Bastien & Romanczuk [12], the controller had two outputs: one to control the speed of the agent, the other to change the heading and with it the direction of movement. For the 3D model one additional output is required to allow movement in all

[2]The resolution used for creation of this plot is the same as for the simulations. Increasing the observation resolution lowers these errors, however, as previously mentioned, the lower resolution was chosen to make the computations faster.

dimensions. The authors proposed two solutions: adding independent velocity in the z axis or adding a second angle which would rotate the velocity vector in the vertical direction. However, since the UAVs allow omnidirectional movement in all three dimensions without changing their heading, it was instead chosen to remove the direction angles completely and instead output a vector describing the required acceleration in each axis of the local frame separately. This approach allows a straightforward way of combining the output with outputs of other controllers by adding the resulting acceleration vectors (see Section 2.4). Furthermore, since no specific heading is required by the controller, the control of heading can be used for another, potentially task specific, purpose.

2.3 Visual controller

The visual controller is a fully distributed stateless controller that runs on each agent. Input to the controller is the visual observation described in Section 2.1. The reference accelerations are calculated using the following equation:

$$\mathbf{a}_{\text{vis}}(\mathbf{V}) = S_{\text{rep}}\mathbf{f}_{\text{rep}}(\mathbf{V}) + S_{\text{att}}\mathbf{f}_{\text{att}}(\mathbf{V}), \tag{2.4}$$

where

$\mathbf{a}_{\text{vis}}(\mathbf{V})$ is 3D vector of output accelerations,
$\mathbf{f}_{\text{rep}}(\mathbf{V}), \mathbf{f}_{\text{att}}(\mathbf{V})$ are functions (set of all observations $\to \mathbb{R}^3$) for calculating the repulsion and attraction forces from the visual input,
S_{rep} is repulsion strength,
S_{att} is attraction strength,
$\mathbf{V}$ is the input observation image.

Both repulsion and attraction forces are calculated directly from the observation image. Since the input does not directly contain any information about the distance to the other agents, the controller has to act only on the observed silhouettes. A geometric property of these shapes that is used to control the agents is that when an agent is closer the resulting silhouette is larger then when the agent is further away. Since the area of a 2D shape grows quadratically with it's radius, but the edge length increases only linearly, the ratio of area to edge length is higher the larger the silhouette is. We can use this property to create the attraction and repulsion forces, by calculating the repulsion from the area and the attraction from edge length as described in the following equations:

$$\mathbf{f}_{\text{rep}}(\mathbf{V}) = \Delta\alpha^2 \sum_{r=0}^{h-1}\sum_{c=0}^{w-1} \mathbf{V}_{r,c}\mathbf{r}_{\text{rep}}(r,c)W(r,c), \tag{2.5}$$

$$\mathbf{f}_{\text{att}}(\mathbf{V}) = \Delta\alpha \sum_{r=0}^{h-1}\sum_{c=0}^{w-1} \text{edge}(\mathbf{V},r,c)\mathbf{r}_{\text{att}}(r,c)W(r,c), \tag{2.6}$$

where

$\mathbf{r}_{\text{att,rep}}(r,c)$ are response functions – functions (set of all pixel coordinates $\to \mathbb{R}^3$) calculating the response to the specified pixel;
$W(r,c)$ is weight of the specified pixel, used to counter the distortion introduced by projecting the spherical observation onto the flat image;
$\mathbf{V}_{r,c}$ is pixel value on the row r and column c, either 0 or 1;

$$\text{edge}(\mathbf{V},r,c) = \begin{cases} 1 & \text{pixel on row } r \text{ and column } c \text{ is 1 and has a neighboring (up, down, left, right; first and last columns are neighbors) pixel with value of zero,} \\ 0 & \text{otherwise;} \end{cases}$$

h, w [px] are height and width of the observation image;
$\Delta\alpha$ is the change of angle between two neighboring pixels.

Both the response functions and the weight function can also be expressed with arguments θ and φ, which are the spherical coordinate angles corresponding to the pixel at coordinates (r, c) as described in Equation (2.2).

For the observation introduced in Section 2.1, the weight function is defined as:

$$W(\theta, \varphi) = \cos(\theta). \tag{2.7}$$

This compensates the stretching of the lines caused by the projection from sphere to rectangle (Equation (2.2)). As a result, the compensated area of the projected silhouettes is independent of their position in the observation image. Since the edges are both horizontal and vertical, it works only as a lower bound approximation for the edge lengths. This approximation is safe to use as it can only result in weaker attraction and thus cannot lead to collisions.

The behavior of the agents is primarily determined by the two response functions, which describe the responses of agents to the individual pixels in the observation image. By choosing different response functions, it is possible to control the resulting properties of the swarm, such as its shape.

Spherical response functions. The first set of response function used for evaluation of the algorithm is:

$$\mathbf{r}_{\text{rep}}(\theta, \varphi) = \begin{bmatrix} -\cos(\varphi)\cos(\theta) \\ -\sin(\varphi)\cos(\theta) \\ -\sin(\theta) \end{bmatrix}, \tag{2.8a}$$

$$\mathbf{r}_{\text{att}}(\theta, \varphi) = -\mathbf{r}_{\text{rep}}(\theta, \varphi), \tag{2.8b}$$

which results in a symmetrical behavior in all three dimensions. These functions are based on transforms from spherical to cartesian coordinates, the resulting forces are colinear with the direction vectors corresponding to the input angles. These response functions are a straightforward extension of the 2D version [12] into 3D.

Planar response functions. Additionally, a second set of response functions was created with as specific swarm shape in mind. Instead of a three dimensional swarm, we wanted to achieve a planar swarm parallel to the xy plane, meaning the agents would align the z coordinate. The response functions for this behavior are defined as:

$$\mathbf{r}_{\text{rep}}(\theta, \varphi) = \begin{bmatrix} -\cos(\varphi) \\ -\sin(\varphi) \\ 0 \end{bmatrix}, \tag{2.9a}$$

$$\mathbf{r}_{\text{att}}(\theta, \varphi) = \begin{bmatrix} \cos(\varphi) \\ \sin(\varphi) \\ \sin(\theta) \end{bmatrix}. \tag{2.9b}$$

The functions for attraction and repulsion forces in the xy plane are based on the transform from cylindrical to cartesian coordinates, similarly to a 2D model, which only works with the xy plane. The z axis the attraction force was chosen to be the same as in the spherical response functions (see Equation (2.8)), but the repulsion was removed completely. The absence of repulsion in the z axis allows the agents to align the z coordinate with others, while the responses in the xy plane are used to both avoid collisions and keep the swarm together.

This approach to planar swarm has a key advantage compared to using the 2D model directly. When using 2D model in a 3D environment, additional controller is required to align the agents to the same plane, because the 2D controller cannot influence movement in the z axis. On the other hand, this set of response functions results in a similar planar swarm, without the need for an added controller that aligns the altitude of individual agents.

2.4 Augmented controller

Section 2.3 described the visual controller itself, however, it is not always sufficient for controlling the agents on its own. Firstly, it is prone to inducing oscillations in the swarm (see Section 2.4.1 for more details); secondly, even when stabilized, the swarm stays near some equilibrium and does not move without some outside influence, making it unable to complete tasks. In this section, we describe some additions that make the swarm more stable, allow position control of the swarm or result in some emergent behavior, such as flocking.

All of the added controllers output the acceleration vector in the same format as the visual controller (see Section 2.2), thus, the combined output can be obtained as a linear combination of the individual outputs:

$$\mathbf{a}_{\text{aug}}(\mathbf{V}, \mathbf{q}) = S_{\text{vis}}\mathbf{a}_{\text{vis}}(\mathbf{V}) + \sum_{i=1}^{N} S_i \mathbf{a}_i(\mathbf{q}), \tag{2.10}$$

where

$\mathbf{a}_{\text{vis}}$ is the visual controller as described in Equation (2.4),
$\mathbf{a}_i \;\; i \in 1, \ldots, N$ are additional controllers described further,
$S_{\text{vis}}, S_i \;\; i \in 1, \ldots, N$ are strengths of the corresponding controllers,
$\mathbf{q}$ is a state containing all of the data required by the added controllers.

The strength coefficients can be used to fine-tune the behavior of the swarm by changing the impact of the individual parts of the augmented controller.

2.4.1 Drag controller

During initial testing in simulations, it was discovered that the visual controller by itself was prone to inducing oscillations instead of stabilizing in an equilibrium, which increased chances of collisions between the agents. Furthermore, the excess movement of agents is not desirable, since it may interfere with a task that should be performed and may lower the flight time of the UAVs. The cause of these oscillations are the chosen control outputs being accelerations. When the agents are near an equilibrium position, the control output is close to zero. However, since the agent is probably moving, it will go through the equilibrium position to the other side, where the controller will start pulling the agent back. This can result in oscillations with increasing magnitude. This problem is similar to controlling position of a double integrator with only a proportional controller, which can be solved by including a derivative term. Analogously to the solution with double integrator, a linear drag was added to the augmented controller, in the form:

$$\mathbf{a}(\mathbf{v}) = -D\mathbf{v}, \tag{2.11}$$

where $\mathbf{v}$ is the velocity of the agent and D is the drag coefficient. This controller forces the agent to slow down and thus suppresses most oscillations. Increasing the drag coefficient D results in less oscillations, however, the agents are slower overall. This also lowers the speed,

with which the agents are able to follow others just using the visual swarm controller, which is especially noticeable in a leader–follower type scenario. Another potential drawback of this addition is that the behavior is now also dependant on the absolute speed of the swarm, not just the relative positions. However, the benefits (mainly stability and safety) of adding the drag controller far outweigh the mentioned drawbacks, hence it was included for all experiments.

2.4.2 Goal controllers

For completing practical tasks, it is also required to be able to control the agent in some other way to achieve a given goal. An example of such goal would be moving the agent to a specified location or in a specified direction. To implement these goal controllers, we used Proportional–Integral–Derivative (PID) controllers that output reference accelerations for the agent. This allowed integrating the goal into the augmented controller.

Depending on the number of agents with the goal controller, the visual swarming algorithm fulfills slightly different roles. When the same goal controller is added to all of the agents, the swarming algorithm has primarily a role similar to a collision avoidance system, while the compactness and alignment of the swarm is primarily influenced by all agents trying to reach the same place. On the other hand, if only some of the agents have the goal controller, the swarming algorithm is also responsible for keeping the agents together, thus creating a leader–follower type scenario.

The goal controllers described in this work were based on some global targets, however, the concept of these goal controllers is much broader. They can cover many tasks required by the application for which the swarm is used. An example of a task could be following a moving target that is relatively localized by onboard sensors. Even when some of the agents lose track of the target, they will still follow the rest of the swarm and start tracking the target again, as soon as they detect it.

2.4.3 Self-propelled movement

Another potential addition to the controller is the introduction of preferred speed of the agent. This self-propelled movement was present in the original approach [12], and was removed in this work together with the change of the outputs from speed and heading to accelerations. With the movement present, it is more challenging to stabilize the swarm in place. The self propelled movement can also interfere with the goal controllers described above. However, the removal of this movement also made it impossible for emergent behaviors, such as flocking, to occur.

To reintroduce this possibility, a controller for holding a desired speed can be added to the augmented controller. It was designed to be able to hold a specified speed of the agents while also affecting the direction as minimally as possible. The speed controller is implemented as a PID controller, with input being the agent's speed (norm of the velocity vector) and the output being a scalar acceleration, which is then applied in the direction of the current

velocity[3]:

$$\begin{aligned} \mathbf{a}_{\text{speed}}(\mathbf{v}) &= x\mathbf{d}, \\ s &= \|\mathbf{v}\|, \\ x &= \text{PID}(s), \\ \mathbf{d} &= \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|} & \|\mathbf{v}\| \neq 0, \\ \text{random unit vector} & \text{otherwise}, \end{cases} \end{aligned} \tag{2.12}$$

where

$\mathbf{v}$ is the velocity of the agents,
PID is a PID controller for controlling the speed.

2.4.4 Limited turning rate

With the change of the visual controller outputs to 3D acceleration vector, the ratio between tangential and normal components of the output acceleration cannot be influenced by changing parameters of the controller. This means that the strengths with which the controller changes speed and direction are coupled and we cannot change one without affecting the other. In the work by Bastien & Romanczuk [12], the emergent flocking behavior occurred when the turning rate was low and the speed modulation was high enough. To be able to reproduce these results with this extended 3D model, it was required to lower the visual controller's acceleration perpendicular to the velocity vector, while keeping the tangential component of the acceleration unchanged. This limitation of turning rate was performed using the following modification of the visual controller:

$$\begin{aligned} \mathbf{a}_{\text{lim}}(\mathbf{V}, \mathbf{v}) &= \mathbf{p} + K\mathbf{r}, \\ \mathbf{P} &= \frac{\mathbf{v}\mathbf{v}^{\mathrm{T}}}{\|\mathbf{v}\|^2}, \\ \mathbf{p} &= \mathbf{P}\,\mathbf{a}_{\text{vis}}(\mathbf{V}), \\ \mathbf{r} &= \mathbf{a}_{\text{vis}}(\mathbf{V}) - \mathbf{p}, \end{aligned} \tag{2.13}$$

where

$\mathbf{a}_{\text{lim}}(\mathbf{V}, \mathbf{v})$ is the output of the visual controller with limited turning,
$\mathbf{V}$ is the observation image,
$\mathbf{v}$ is the velocity of the agent,
$K \geq 0$ is the turning strength ($0 \sim$ no turning possible, $1 \sim$ normal turning rate).

This controller is used as a replacement for the visual controller $\mathbf{a}_{\text{vis}}(\mathbf{V})$ in the augmented controller (Equation (2.10)). By changing the parameter K we can configure the turning strength of the visual controller. This limitation was only active if the agent was moving with a speed greater than 0.1 m/s to not interfere with the movement of almost stationary agents.

2.4.5 Wall repulsion

For some experiments, an environment with walls was tested. This was primarily used to study the effects of interactions with environment when testing the emergent behaviors.

[3] From this point of view, the drag controller (introduced in Section 2.4.1) can be though of as a speed holding proportional controller with a target speed of 0 m/s.

The wall avoidance controller was implemented using a repulsive field around the wall. The repulsion force was perpendicular to the wall, reaching 15 m from the wall and increased linearly as the agent got closer. For the wall avoidance controller, all of these forces were added together and outputted as 3D accelerations for the agent. Thanks to this choice, it was not required to pause the other controllers during avoidance of the wall, as the outputs were combined together in the augmented controller (Equation (2.10)).

3 Implementation

3.1 Implementation of the swarm controller

The augmented swarm controller, as described in Section 2.4, was implemented in the C++ programming language as an agent controller for a custom simulator that was developed for this project (see Section 4.1). Subsequently, the controller was ported to work as a controller inside the MRS UAV System [14], which is built on top of Robot Operating System (ROS) [15]. This port of the controller allows interactions with both the MRS Multirotor Simulator [16] as well as real UAVs.

3.1.1 Simulating observations

Since the observation system is not the focus of this work, we did not create a visual system for creating the observations. Instead, we created a simulated system capable of generating the required observation. Observations were simulated by constructing the images from positions of the other agents and their known sizes. Depending on the setup, different sources of positions were used to calculate the observations. These sources are described in the sections about simulators (Sections 4.1 and 4.2) and real-world deployment (Section 3.2). Using the positions of all other agents, the observations were created according to the Equation (2.3), where the detection function $V(\theta, \varphi)$ (Eq. (2.1)) was implemented using a check for ray-sphere intersection. In all cases, the positions of other agents were only used to calculate the observations and were not used by any part of the controller.

3.1.2 Blind spots

In addition to simulating the unrestricted observations, we also experimented with limiting the FOV of the simulated detection system. This was achieved by filtering the positions such that only agents inside the FOV were drawn onto the observation and the rest of the agents was ignored. This was done to both test the limitations of the algorithm as well as prepare for real-world experiment, where we planned to use the UltraViolet Direction And Ranging (UVDAR) system [17], [18], which would have had some blind spots.

Vertical blind spots. First blind spot pattern limited only the vertical FOV, where only the agents corresponding to angles $\theta \in \langle -\pi/4, \pi/4 \rangle$ were kept and the rest of them were ignored. This was done to simulate detection systems that do not have full coverage of the 3D space around the agent, but rather only detect agents at a similar altitude (e.g., horizontally mounted cameras, Light Detection and Ranging (LiDAR)).

UVDAR blind spots. The second tested blind spots pattern was designed to simulate the expected blind spots of the UVDAR system. This meant using the vertical blind spots described above and adding 20° blind spots on both sides, corresponding to the angles $\varphi \in \langle 1.40, 1.75 \rangle$ and $\varphi \in \langle 4.54, 4.89 \rangle$[1].

[1] $\langle 80°, 100° \rangle$ and $\langle 260°, 280° \rangle$ respectively.

3.1.3 Mitigation of blind spots

Since the swarming algorithm works independently of the controlled agent's heading, we can use the heading for different purposes, such as scanning the surroundings with the limited FOV cameras. By rotating the heading constantly, the agent moves the horizontal blind spots around, dramatically lowering the chance of others staying in the unobservable area. This behavior does not interfere with the swarm controller and can be integrated seamlessly alongside it.

3.2 Real-world setup

Even though the visual swarming algorithm itself needed no change for deployment in the real world, some of the accompanying parts of the controllers had to be modified. This section describes all the modifications that were necessary for the real-world experiments with the swarm controller.

3.2.1 Drift mitigation

Since the swarm controller is outputting reference accelerations, it is prone to drifting from position, which can result in the UAVs flying away or crashing into the ground. To counteract this issue during the real-world experiments, the following controllers were used as a part of the augmented swarm controller:

- **Altitude band.** This controller was responsible for keeping the UAV in a safe altitude band. It was implemented using a PID controller with error being the distance from the chosen altitude band (0 when inside the band). This allowed for stopping the vertical drift of the UAV as well as compensating for an inaccurate mass estimate, which otherwise caused the UAV to quickly lose or gain altitude. Since the controller was only keeping the UAV in the chosen altitude band, it did not interfere with the vertical movement inside this band requested by other controllers.
- **Target.** The second controller was used to move the UAV to chosen coordinates and keep it there. It was implemented using a PID controller similar to the one used for goal controllers in the simulation, but with stronger integral term to be able to hold the target position even in a moderately strong wind. This controller was disabled on some of the UAVs during the flight to test a leader–follower scenario.

3.2.2 FADAR

Originally, we planned to use a new version of the UVDAR system for relative localization of other agents. However, the authors of the UVDAR module have not yet finished the system at the time of the experiments, so we had to obtain the observations from a different source.

For this reason, a simulated version of the UVDAR system was created, which we named Fake Detection and Ranging (FADAR). This system uses a ROS package called `nimbro_network` [19] to transport the required data over a Wi-Fi network. Instead of using UVDAR, each agent is running an instance of the FADAR node, which sends the current position of this agent to others, collects the positions of all other agents and wraps them into an interface compatible with the UVDAR interface. This allows switching between UVDAR and FADAR just by launching the chosen detection node, without the need to modify any code or configurations of the controller.

During the real-world experiments, the agents used Real-time Kinematics (RTK) Global Positioning System (GPS) to get the position estimates and shared them at a frequency of 10 Hz using the FADAR system. These positions were then used to calculate the simulated observations in the same way as in the simulations, allowing real-world evaluation of the swarming algorithm without the need for a detection system for creating observations.

4 Evaluation methodology

4.1 Custom swarm simulator

For majority of the experiments, a custom swarm simulator based on Euler method was used. This simulator only takes into account the positions, velocity and accelerations of the individual agents (3D double integrator physics)[1], and allows for fast simulations with larger swarms at the cost of some accuracy. The simulator has configurable limits for maximal speed and acceleration of the agents, but there are no limits for jerk.

The swarm controllers of the individual agents interact directly with the simulator, which means the controllers get the exact data from the simulator and the calculated acceleration output is directly applied in the physics engine. There are multiple options for creating the observations. Firstly, we can specify whether to directly use the position data provided by the simulator or to first add a noise by randomly altering the positions. Secondly, a position filter can be added to simulate the blind spots described above. Lastly, a maximal range of detection can be set, which causes the observations to ignore any agents further than the specified distance.

This simulator also provides the option to set the frequency at which the controllers and the physics simulation steps are executed. For all of the simulations described later, this simulator was configured to run the controllers with a frequency of 30 Hz and the physics simulation at 120 Hz.

4.2 MRS Multirotor Simulator

A second simulator used for the evaluation of the algorithm was the MRS Multirotor Simulator [16]. It simulates the full dynamics of the UAVs and uses MRS UAV System [14] to control the UAVs.

Firstly, the simulated observations of individual agents were calculated from ground truth positions (later with added noise) of other agents provided by the simulator. This allowed assessing, whether the introduction of higher order dynamics significantly affects results.

After that, a more realistic method for obtaining the observations was introduced. This was done by using positions of other drones estimated by a simulated UVDAR [17], [18] system[2]. The final observations used by the controller were still calculated from these estimated positions, however, the estimates themselves were created by a simulated visual system instead of ground truth data provided by the simulator and therefore they include realistic noise. The UVDAR system was chosen because we planned on using it during real-world experiments.

4.3 Metrics

The results of experiments (simulated and real-world) were evaluated both qualitatively and quantitatively. Firstly, a qualitative evaluation was performed, observing results such as the overall behavior, shape of the swarm, whether a goal was reached, etc. If the results were unsatisfactory (e.g., there were collisions due to wrong parameters of the model), the

[1]The custom simulator simulator does not take into account the heading of the individual agents, as well as any tilt required to accelerate the UAV in a horizontal direction.

[2]UVDAR simulation used: [20].

experiment was considered failed and was only analyzed to find the best ways to mitigate the current issue. Secondly, if the results passed the qualitative evaluation, the behavior was quantified using metrics described in this section. These metrics were chosen to give deeper insight into the behavior of the swarm and to allow comparisons between individual experiments.

4.3.1 Distances

First of the evaluated properties of the swarm are the mutual distances between agents. For this, we used two metrics:

(i) minimal distance:

$$d_{\min} = \min_{i \in I, j \in I, i \neq j} \{d(i,j)\}, \tag{4.1}$$

(ii) average minimal distance:

$$d_{\text{avgmin}} = \frac{1}{|I|} \sum_{i \in I} \min_{j \in I, i \neq j} \{d(i,j)\}, \tag{4.2}$$

where

I is the set of all agent's indexes,
$d(i,j)$ is the Euclidean distance between i^{th} and j^{th} agent.

The target behavior is to maintain some safe distance between all agents that does not change. In the context of these distances, it translates to the metrics not changing much over time and the values of both metrics being close to each other, while staying above a chosen safety threshold (which was chosen to be 3 m).

4.3.2 Polarization

Next metric for evaluating the results is called polarization (or alignment) of the swarm. It shows how aligned are the directions of movement of the individual agents. A similar metric was used for evaluation of results in the work by Mezey et al. [13] and we adapted it for use for our 3D model as follows:

$$P = \frac{1}{|I|} \left\| \sum_{i \in I} \hat{\mathbf{v}}_i \right\|, \tag{4.3}$$

where

I is the set of all agent's indexes,
$\hat{\mathbf{v}}_i$ is the i^{th} agent's movement direction (normalized velocity vector, zero vector if the velocity is zero).

The values of this metric range from 0 to 1, with 1 meaning all agents are moving in the exact same direction (regardless of speed) and numbers close to 0 meaning the agents are moving in random or close to opposite directions.

This metric is very important during parts of the experiments, when the whole swarm is moving, since it provides information on how well the agents are able to align their velocities with the rest of the swarm. However, in case of stationary swarm, value of this metric is usually random or close to zero, since there is no reason for the agents to move with aligned velocities.

4.3.3 Swarm velocity

Last of the presented metrics are the velocity of the swarm:

$$\mathbf{v}_{\text{swarm}} = \frac{1}{|I|} \sum_{i \in I} \mathbf{v}_i \tag{4.4}$$

and the swarm speed:

$$v = \|\mathbf{v}_{\text{swarm}}\| \,. \tag{4.5}$$

These velocity metrics allow evaluation of speeds at which the controller is able to operate as well as potential changes in behavior caused by different speeds of the swarm. Furthermore, from the swarm velocity, we can observe the impacts of the controller on the direction of movement of the swarm (e.g., whether the has a tendency to influence the direction of movement). It is also important for interpretation of the polarization P, since it becomes mostly irrelevant during low speeds of the swarm.

5 Results

Most of the testing and evaluation of the algorithm was performed in simulated unconstrained environment, i.e., environment, that is infinite in all directions and without any obstacles. For initial evaluation of the individual scenarios, the custom simulator was the best choice. It allows effective tuning of the parameters, since the simulations are much faster compared to the MRS Multirotor Simulator. The interpretation of the obtained results is more straightforward, since they are not influenced by the dynamics of a real UAV and more directly show actions of the evaluated controller.

5.1 Parametrization

Firstly a set of viable parameters S_{rep}, S_{att} and D was found, such that the agents were always keeping some minimal safe distance from others. Secondary criterion was that the swarm should be as compact as possible. Finding optimal parameters is beyond the scope of this project, thus only some viable parameters were found and used for further testing.

The parameters we found during the initial testing were not suitable for all simulated scenarios, e.g., setting limit on the observation range required changing the parameters to keep similar distance to nearest agent (see Section 5.5.1). As a result, we used different sets of parameters during the tests in the custom simulator, MRS Multirotor Simulator and during the real world experiments. The parameters used in the presented experiments are shown in Appendix A.

5.2 Simulated scenarios – custom simulator

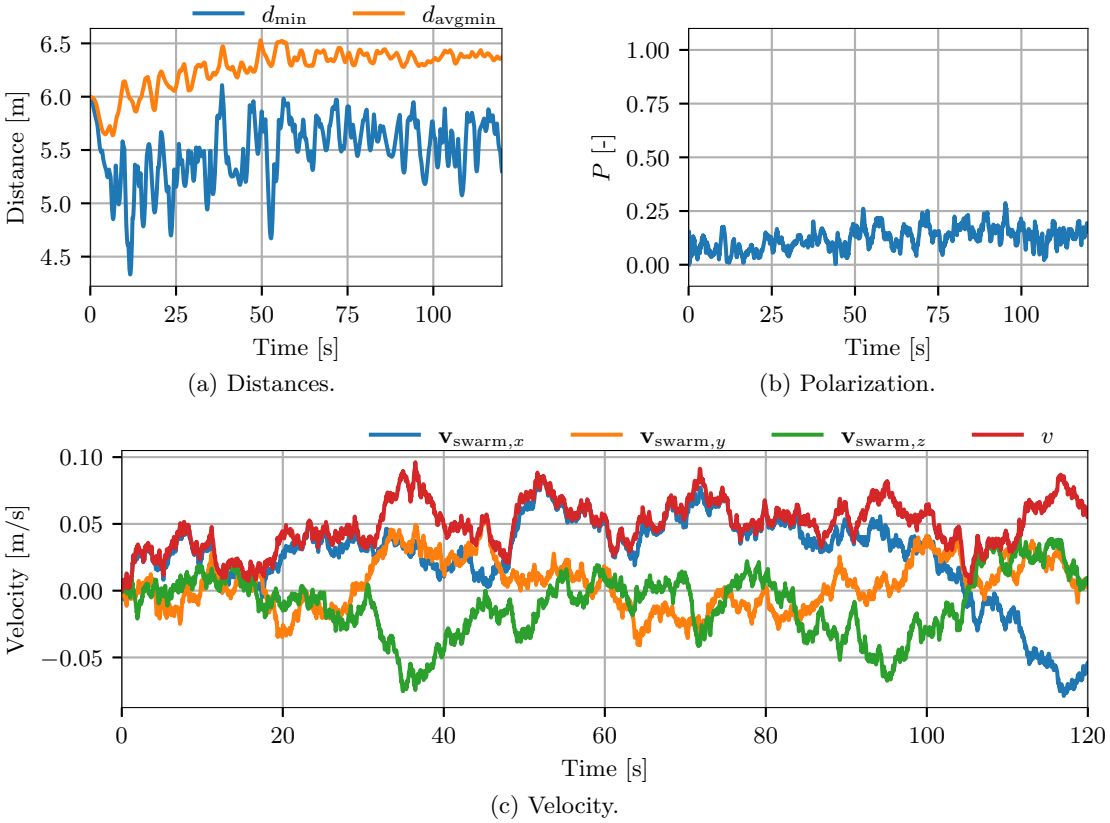
This section presents results from the custom simulator. Since many different experiments were conducted, we present only the most important results that best represent the overall behaviors that were observed throughout the experiments. All scenarios listed in this section were performed with at least 20 agents. The results we present are using observations with simulated noise. For the analysis of the impact of the noise, see Section 5.3.

5.2.1 No goal

First of the tested scenarios was a baseline simulation with no goal for the agents. This setup was used to find parameters used for most of the experiments. All sets of viable parameters that were found during the evaluation resulted in the swarm converging to a crystal-like structure with agents oscillating around mostly constant positions. Depending on the response functions set that was used, the swarm's structure was either three dimensional (spherical response functions, Eq. (2.8)), or two dimensional, parallel to the xy plane (planar response functions, Eq. (2.9)). In both cases, the shape of the whole swarm tended to approach ball or disk respectively.

The search for parameters was done by fixing the repulsion strength S_{rep} and changing the attraction force S_{att} to keep the desired distances. The drag coefficient D was set to a low number and only increased if the swarm destabilized itself by inducing strong oscillations.

This scenario was simulated with with many combinations of parameters and agent counts. Figure 5.1 shows metrics for a simulation with the finalized parameters that were



(a) Distances.

(b) Polarization.

(c) Velocity.

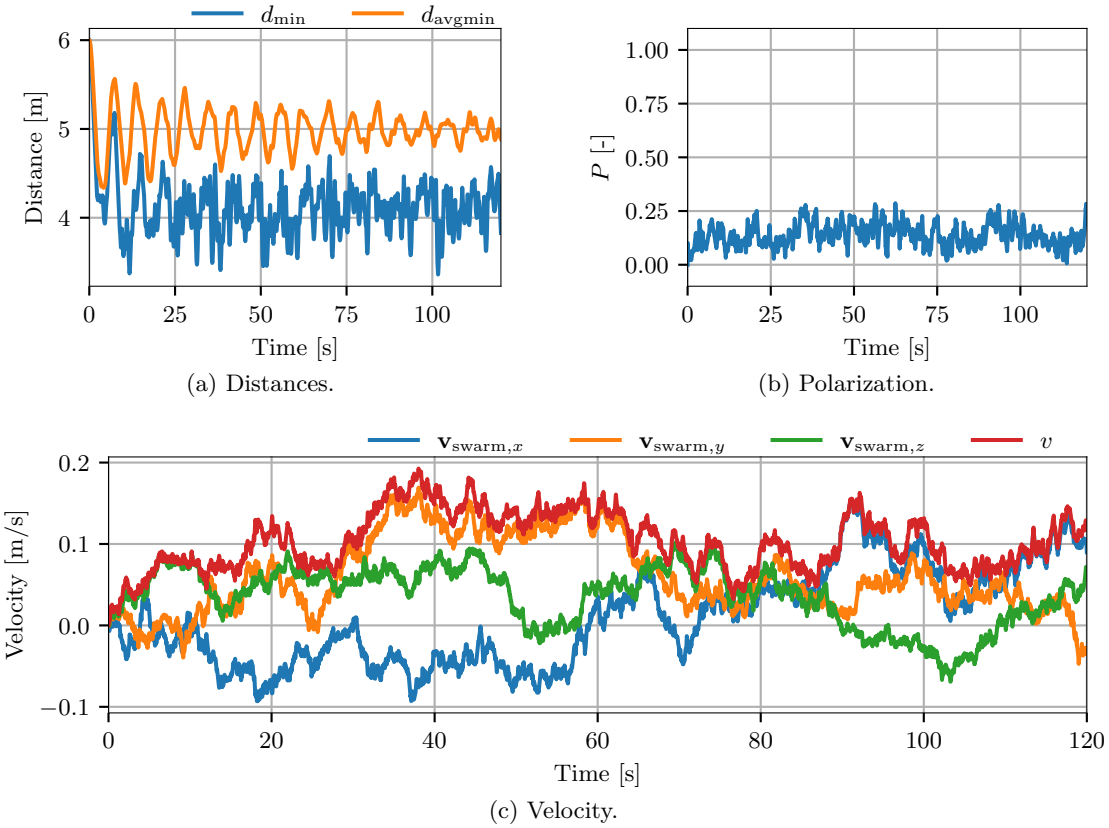
Figure 5.1: Metrics for the no goal scenario – 27 agents in a $3 \times 3 \times 3$ initial configuration, spherical response functions (Eq. (2.8)).

also used for all the other experiments in this section. This simulation was performed with 27 agents initially spaced 6 m apart in a $3 \times 3 \times 3$ grid, using the spherical response functions (Eq. (2.8)). Figure 5.1a shows that the average minimal distance was around 6–6.5 m during the majority of the experiment. The minimal distance was less consistent, but overall has never gotten close to the chosen minimal safe distance of 3 m. The polarization metric (Figure 5.1b) is low during the whole simulation, which is to be expected, since the whole swarm was mostly stationary (see velocities in Figure 5.1c) and most of the movement was due to oscillations, movement around the center of the swarm and input noise.

The same experiment was also performed with the planar response functions (Eq. (2.9)), and a starting grid of 5×5 agents spaced 6 m apart. The resulting polarization (Figure 5.2b) and velocities (Figure 5.2c) are similar. The main difference compared to the spherical response functions is that with the same parameters, the agents with planar response functions are closer, with the average minimal distance around 5 m (see Figure 5.2a) compared to the 6–6.5 m with spherical response functions.

5.2.2 Shared goal

After the viable parameters were found, we tested other scenarios to evaluate the performance of the algorithm under different conditions. First of these scenarios was a shared goal scenario with all agents having the same goal, which was either to move in a specified direction or move to a specific point. These goals were handled by enabling the goal controller (see Section 2.4.2) on all of the agents.



(a) Distances.

(b) Polarization.

(c) Velocity.

Figure 5.2: Metrics for the no goal scenario – 25 agents in a 5×5 initial configuration, planar response functions (Eq. (2.9)).

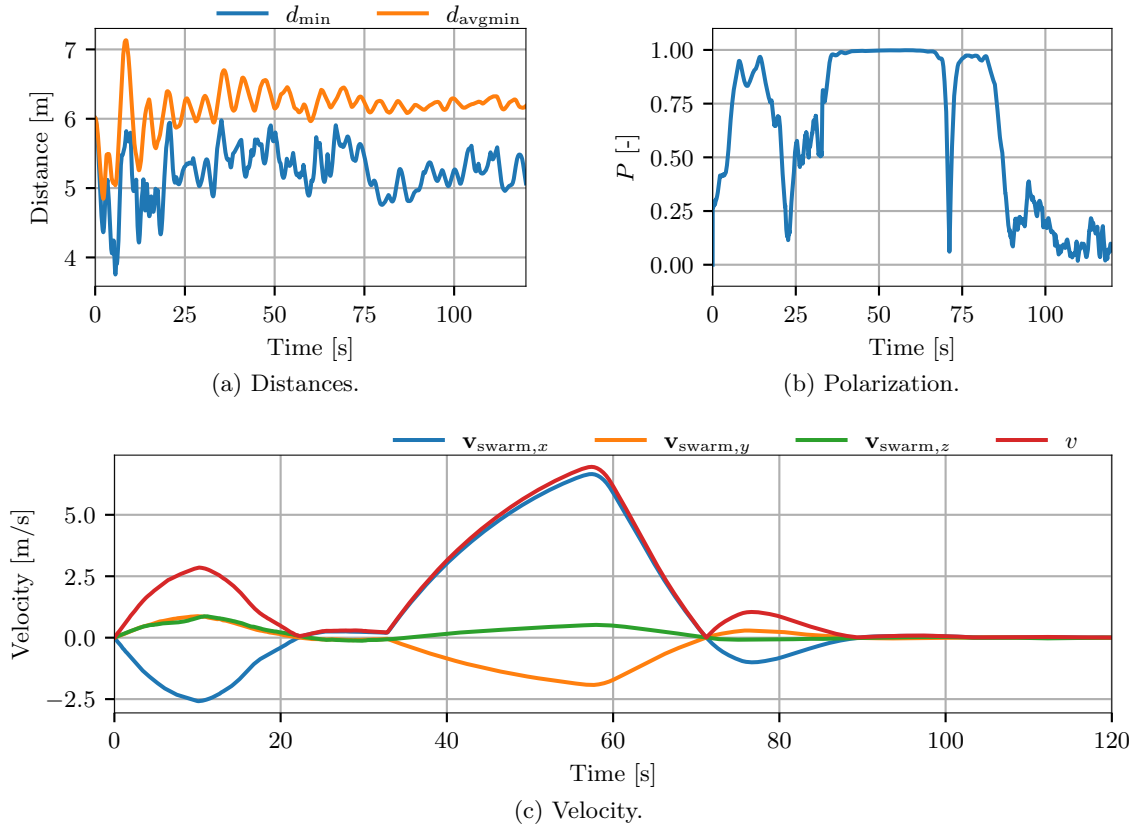
Enabling the goal controller resulted in the swarm moving towards the target as a cohesive group. The shape of the swarm was not affected by the movement and stayed very similar to the previous scenario throughout the experiment. When the goal of all agents was the exact same target coordinates, no agent was usually able to get to the exact position chosen. However, the swarm as a whole was able to stabilize around this position, with center of mass of the swarm staying close to the target position (usually closer than 0.5 m).

Figure 5.3 shows the metrics for a simulation with 25 agents using the spherical response functions (Eq. (2.8)), initially placed in a 5×5 grid, 6 m apart. In this exact simulations, all of the agents had a same goal and once the center of mass of the swarm was close to the desired position[1], the goal for all agents was changed. The change of target occurred around the time of 33 s and is clearly visible in the velocity plot (Figure 5.3c) as a sudden increase in the speed of the swarm.

Since the initial configuration was not stable, the agents started flying up or down to make a sphere shaped group. This fast change of shape resulted in the initial fluctuations in the average distances (Figure 5.3a). After the initial changes, the distances stabilized around the same values as in the no goal scenario, even though the target controller was trying to move all the agents into the exact same spot.

The polarization during movement (Figure 5.3b) was close to the maximal, which is

[1]The center of mass had to be within 0.5 m of the target and the swarm speed had to be lower than 0.5 m/s.



(a) Distances.

(b) Polarization.

(c) Velocity.

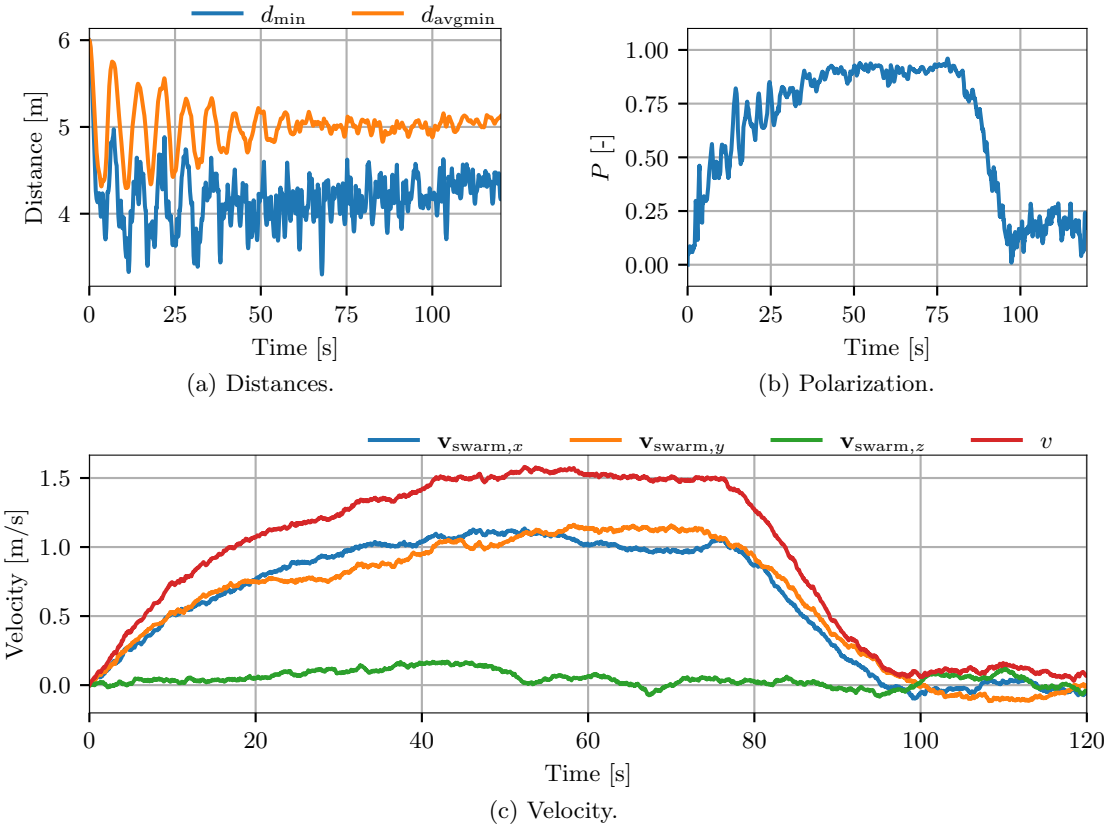
Figure 5.3: Metrics for the same target scenario – 25 agents in a 5×5 initial configuration, spherical response functions (Eq. (2.8)).

mostly the result of all agents having the same target. In this case, the swarm controller is not responsible for this alignment of velocities. In the first part, the polarization is lower because the agents were still reconfiguring from the unstable initial conditions and thus they were not moving only towards the target. The sudden dips in polarization around 23 s and 70 s were caused by the swarm overshooting the target, stopping and starting to move in the opposite direction. After the second goal is reached (around 86 s) the polarization goes down quickly for the same reasons as in the no goal scenario.

5.2.3 Leader(s)–followers

The next scenario was a leader follower setup, with one or more agents sharing a common goal (leaders) and the rest of the agents without any goal (followers). The results of this scenario were influenced mainly by the following parameters:

- **Number of leaders.** When only a single agent had a goal set, the swarm moved slower and there was a higher chance of the leader splitting from the rest of the swarm and flying away. With increasing number of leaders, the swarm was able to move faster and had a lower chance of splitting. The stability was further increased when the leaders were spread among the other agents as opposed to being clumped together.
- **Strength of the goal controller.** Maximal strength of the goal controller was an important parameter for this task, as it greatly affected the resulting behavior. With low strength, the swarm moved slowly, as the goal controller was unable to overcome the



(a) Distances.

(b) Polarization.

(c) Velocity.

Figure 5.4: Metrics for the leaders–followers scenario – 25 agents (4 of which are leaders) in a 5×5 initial configuration, planar response functions (Eq. (2.9)).

drag force of the followers' controllers. Increasing the strength up to a certain point resulted in an increase of the swarm's speed, however, there was a limit, at which, the vision controller was no longer able to keep the swarm together and the leaders split from the rest of the swarm.

- **Drag coefficient.** The drag controller parameter D had a large impact on the resulting speed of the swarm. Since the leaders have to overcome not only the drag of their own controllers, but also the drag of all of the followers, lowering the drag coefficient D significantly increased the maximum speed the swarm was able to reach. For this reason, it is beneficial to choose the lowest possible D parameter that is able to suppress the buildup of oscillations in the swarm.

As in the previous cases, this scenario was tested with both sets of response functions, both of which mostly kept their characteristic shapes. However, the swarm tended to stretch in the direction of travel, as the leaders moved through the swarm and ended up pulling the rest of the swarm behind them. This pulling resulted in slightly higher than normal distances between the agents in the direction of travel, while mostly not affecting the distances perpendicular to the direction of travel. This effect was more noticeable when the range of detection for the visual observation was limited.

Figure 5.4 shows the measured metrics for the leader–follower scenario with 4 leaders and 21 followers, initially placed into a 5×5 grid, 6 m apart. This simulation was performed using the planar response functions (Eq. (2.9)).

Figure 5.4a shows that the distances between the agents once again stayed similar to the ones during the simulation of the no goal scenario. All of the agents stayed clear of the safety area of 3 m around agents.

The most important metric in this case is the polarization shown in Figure 5.4b. Compared to the results of the shared goal scenario, the polarization is slightly lower, however, in this case, only 4 of the agents know the target and the remaining 21 are following only thanks to the swarm controller. This clearly shows that the visual controller is able to keep the swarm coherent and align the movement of the agents, despite some of them moving away from the rest of the swarm.

Finally, the speed (shown in Figure 5.4c) is lower than what was achieved during the shared target simulation, peaking at around 1.5 m/s. This is to be expected since the target controllers of the leaders have to push against the drag controllers of all of the followers. The drop in velocity and alignment at around 80–90 s is caused by the leaders reaching their goal, and stopping in its vicinity together with the rest of the swarm.

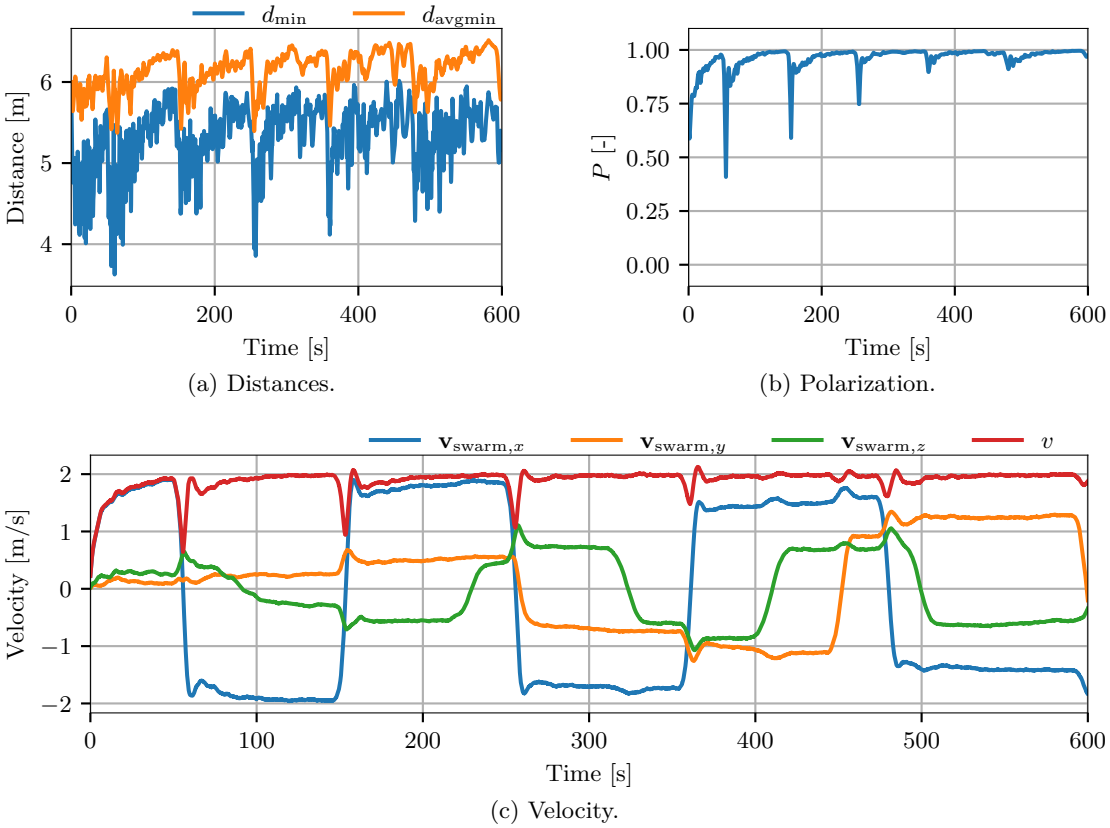
5.2.4 Emergent behavior

The aim of this experiment was to observe the emergent behaviors created by the visual controller. For these experiments, we used predominantly the spherical response functions (Equation (2.8)) to evaluate, how the interactions in the added third dimension affect the resulting behavior. For the emergent behaviors to appear, it was necessary to reintroduce the self-propelled movement, as described in Section 2.4.3. With the introduction of this motion, most agents started orbiting around the center of mass of the swarm, however, the swarm as a whole was staying in mostly the same place. This result is similar to the milling behavior described in [13].

In order to achieve the flocking behavior observed in the 2D case [13], the turning rate limiter described in Section 2.4.4 was introduced. With this, the agents were no longer able to keep orbiting around the center and the whole swarm was forced to start moving. Depending on the velocities at the start of the simulation, the agents either quickly split (opposite direction, speed too hight to be able to stop) or aligned their velocity vectors (most of the cases, where they were not moving too fast and/or the initial directions were similar). After aligning their velocities, all the agents continued moving in the direction they chose and only changed it when they were forced to by external factors (e.g., interactions with walls). This behavior is similar to the one observed in the 2D case [13], where after lowering the ability to turn, the resulting behavior changed to flocking in a direction chosen by the agents.

The same emergent flocking experiment was also tested in an environment constrained by cube shaped walls (see Section 2.4.5 for implementation of the wall repulsion). In this environment, the initial positions of the agents were set to the middle of the cube, and thus the agents had time to align their velocities and start flying in some direction. After reaching a wall, the front of the swarm started slowing down / turning due to the repulsions of the wall. This external force briefly broke the alignment of the movement, however, the agents quickly found a new direction away from the wall, aligned their velocities and continued flying in the new direction. As long as the velocity was not too high compared to the strength of the visual controller, the agents were all able to align themselves after this interaction and none of them split from the group. Continuing the simulation, all of the agents were going from wall to wall, changing direction every time there was some obstacle in the way.

Metrics for the experiment in constrained environment are shown in Figure 5.5. Analyzing



(a) Distances.

(b) Polarization.

(c) Velocity.

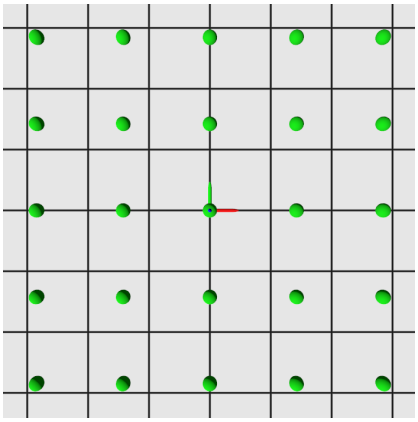
Figure 5.5: Metrics for the emergent behavior scenario – 27 agents in a $3 \times 3 \times 3$ initial configuration, spherical response functions (Eq. (2.8)).

the distances between agents, we can see that they are similar to all of the previous scenarios. An interesting result of this simulation is that during the emergent flocking, the polarization (Figure 5.5b) is able to reach much higher values compared to the leader–follower setup and almost as good as with the shared target. This effect is most likely caused by the speed controllers helping to keep the same speed and the swarming algorithm mainly impacting the direction of movement. The periodic drops in alignment (Figure 5.5b) and mutual distances (Figure 5.5a), together with the changes in velocity (Figure 5.5c) are caused by repulsion from the wall, when the swarm was going almost directly against the wall.

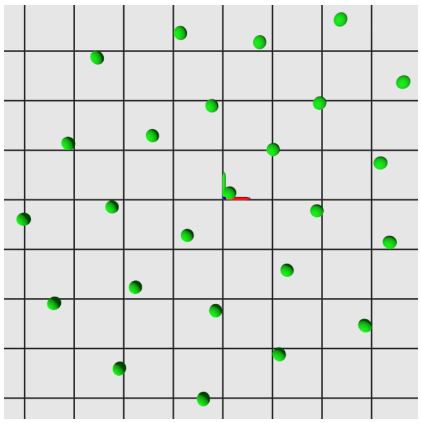
5.3 Robustness

All the scenarios above were tested both with ground truth data and with a simulated noisy observation. Even though no explicit filtering of the input data is done by the algorithm, the effects of the noise on the resulting behavior were minimal. Stability was affected only marginally and the results of the scenarios with noise closely matched their counterparts without the noise (the presented results from the custom simulator were with the noise enabled). Main difference in the results was that the noise broke the initial shape of the swarm in all of the experiments, whereas when no noise was present, this was not the case. In some of the experiments using the ground truth data, we found agents that were unable to leave the initial unstable equilibrium.

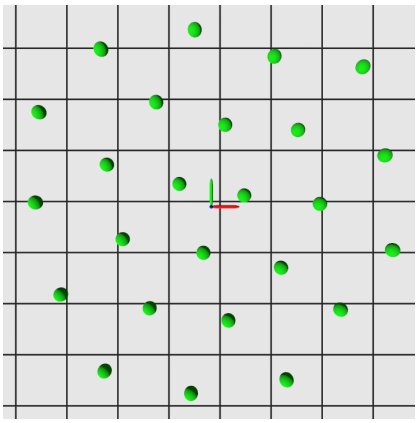
Maintaining these equilibria was only possible in the simulations with no noise, as they



(a) Initial positions for both simulations.



(b) End of simulation without noise.



(c) End of simulation with noise.

Figure 5.6: Comparison of the final positions in a leaders/followers simulation with and without observation noise. 4 leaders, 21 followers, using the planar response functions (Eq. (2.9)).

were usually result of agents starting the experiment on the exact line of symmetry of the swarm. In these cases, the agents were sometimes unable to move out of this line because the interactions with agents on both sides exactly cancelled out. In some experiments, it was observed that these equilibria were making the results worse as the agents were unable to move in certain directions. Consequently, these agents could not move away from some incoming neighbors, resulting in some violations of the safety area around the agents. Running the same experiments repeatedly with the same parameters and enabled noise did not result in any violations of the safety area, thus we can conclude that the observed violations were directly caused by the perfect symmetry disallowing movement in some directions.

The preservation of initial conditions can be seen in Figure 5.6, where a simulations of 4 leaders and 21 followers[2] were simulated for 240 s both with and without the noise. At the start of both simulations, the agents were placed in a 5×5 grid (Figure 5.6a). Figure 5.6b shows the final positions of the agents after the simulation without noise. The agents are still in an axially symmetrical configuration and all five agents at the top-right to bottom-left diagonal in the image (this one was parallel to the direction of travel) are still on the same line as they were at the start of the simulation. After the introduction of noise, these agents did not stay on the line and in some cases, they even moved out of the center and were replaced by different

[2]Planar response functions were used, see Equation (2.9).

nearby agents. The final configuration after the simulation with noise is still a compact swarm, but the axial symmetry of the initial configuration is no longer present (see Figure 5.6c).

5.4 Limited FOV observations

5.4.1 Vertical blind spots

After evaluating the algorithm with the unrestricted observations, we also tested the observations with limited FOV, starting with the vertical blind spots (see Section 3.1.2). With the spherical response functions (Equation (2.8)), the swarm was not stable. Agents quickly lost visual contact with others that went above or below them and subsequently, they either went back to the same altitude and collided, or flew away, not interacting any further. This behavior is to be expected, since the agents have no information about neighbors above and below them. Furthermore, when two agents in similar positions do not see each other, but their observations contain the same neighboring agents, the visual controllers of both of these agents may drive the agents into the same point, further increasing the chances of collision.

This blind spot setup is one of the reasons, why the planar response functions were created (Equation (2.9)). As described above, these functions were designed to align the z coordinate of all agents, which in this situation results in keeping all the agents in the observable region of each other. When all agents started at nearly the same altitude (no agents were out of / close to the edge of the observable region of other agents) the swarm was likely to align the z coordinates of all agents. After the swarm stabilized, the agents did not go out of the observation regions of each other.

This setup was further tested in a leader follower scenario where the leader was instructed to change altitude (go in the direction toward the blind spots of the followers) and the swarm was able to keep up and not lose sight of the leader. Since there is only attraction in the z direction, a situation, where the leader would fly into the unobservable zone was not encountered during testing even with stronger goal controllers.

5.4.2 UVDAR blind spots

The planar response functions were also tested with the UVDAR blind spots (see Section 3.1.2). Without any modification, the controller was not able to stabilize the swarm, because there were often pairs of agents, which were not able to see each other and were forced by their respective controllers to the same or similar position, resulting in collisions.

To mitigate this issue, the constant heading rate described in Section 3.1.3 was introduced. This allowed the agents to better observe their surroundings by moving the blind spots. With a sufficiently fast rotation, the controller was able to stabilize the swarm and the observed behavior was similar to the one during simulations without blind spots. However, even though the swarm controller stabilized the swarm, it still induced more oscillations than with the full horizontal FOV. This was caused by the surrounding agents not being visible all the time.

5.5 MRS Multirotor Simulator

After successful tests in the custom simulator, the no goal, same goal and leaders–followers scenarios were also tested in MRS Multirotor Simulator [16]. This simulator was used primarily to evaluate the performance of the controller with a more realistic physics simulation and to prepare the controller for real-world experiments.

The evaluated scenarios were first tested with observations simulated from ground truth data provided by the simulator. These tests proved that the controller was able to perform well even with the full dynamics of a UAV. Furthermore, the results were mostly similar to the ones observed in the custom simulator. After successful tests using the ground truth data, the simulated UVDAR was added, to evaluate the algorithm with more realistic detection method.

5.5.1 Limited range of detection

Compared to the ground truth data, the UVDAR system has one key drawback – detection range. The real world maximum distance for reliable detection by the UVDAR system is around 15 m [18], which is implemented in the simulation by ignoring all agents further than 15 m [20]. With this limitation, using the swarm controller with the same parameters as in the previous simulations often resulted in the agents spreading too far apart to be able to detect each other. After this happened, the agents could not interact any further and thus the swarm broke apart. This behavior was also confirmed in the custom simulator by limiting the detection range to 15 m.

The root of this issue lies in the fact that the attraction to repulsion force ratio is higher, the further the agent is, and thus, when limiting the observation range, the attraction to further away parts of the swarm is lost, but the repulsion by neighbors is still the same. Furthermore, when the distances between agents oscillate, if the furthest distance is greater than the detection range, the agents stop interacting and the controller is unable to bring them back together. To overcome this issue the attraction strength S_{att} was increased slightly, which made the swarm more compact and less prone to tearing apart.

With the new set of parameters, the swarm was able to perform as expected, although it was still more susceptible to tearing apart than the version with larger or even infinite detection range. In the leader follower scenario, the limited visual range and the resulting tendency to tear the swarm apart also had to be addressed by increasing the number of leaders, lowering the strength of the goal controllers or increasing the strength of the swarm controller. The behavior also became more localized, which resulted in the overall shape of the swarm no longer converging to the disk- or sphere-like shape as it did when there were no limitations on the observation.

Despite the above mentioned drawbacks, the limitation of detection range also had a positive impact on the swarm. In the custom simulator, much larger swarms were tested and the results were more stable when the observation range was limited. When the number of agents was increased to hundreds without limiting the detection range the agents in the center of the swarm were pushed much closer together, often resulting in collisions. However, when limited observation range was introduced, this effect became much weaker and the swarm was able to pass the simulation without any collisions. The reason for this behavior is the previously mentioned change from global to local interactions. When the agents are able to observe the distant agents, they are attracted to them. This happens on all edges of the swarm, pushing toward the center, and thus lowering the distances between agents in the middle. When the swarm is large enough, this results in collisions in the middle of the swarm. The same problem did not arise in the scenario with limited observation range, because the distant agents were not observed, and thus the attraction forces on the edges towards the center were weaker.

5.5.2 Results of tested scenarios

Even with the more complex dynamics simulated in the MRS Multirotor Simulator and the limitations of the simulated UVDAR system, the algorithm was still able to perform as expected. Main difference compared to the more simplified simulation was that the UAVs were less agile in the MRS Multirotor Simulator. The main causes of this behavior are stricter dynamic constraints on the reference accelerations and the delayed reaction to the controller, which is caused by the UAV first needing to tilt before it can accelerate horizontally. This difference resulted in a need to slightly modify some parameters of the algorithm (mainly drag D and strength of the goal controllers) to work better in this environment. Without the modification, the swarm was prone to inducing oscillations and tearing apart. After this change, all of the tested scenarios performed similarly to their counterparts in the custom simulator.

Tests in the MRS Multirotor Simulator were performed with only 10 agents, because of much higher computational performance requirements for running the simulation, together with the MRS UAV System for each of the simulated agents.

5.6 Real-world experiments

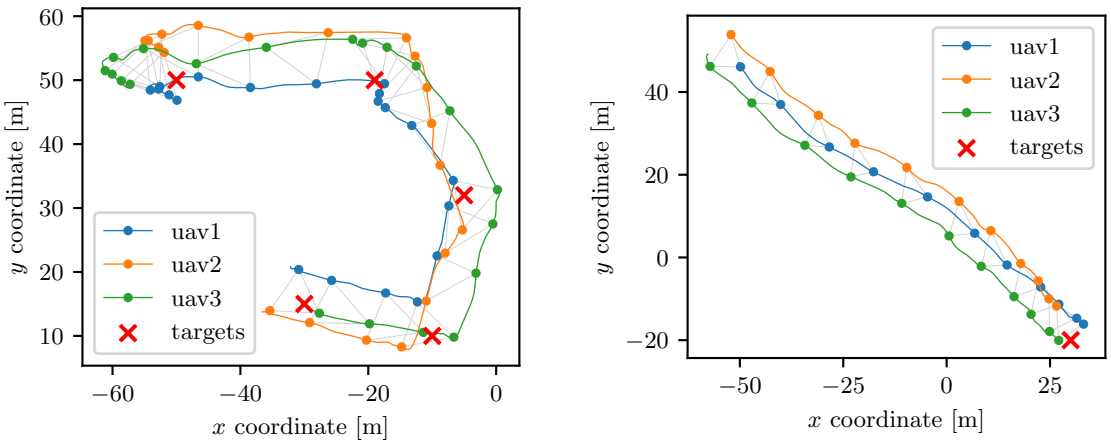
In total, three real-world multi-UAV experiments were performed. All of these experiments had the same startup procedure, during which the UAVs first took off, and flew to a set of predefined positions. After that, the swarm controller was activated, setting the target controllers (see Section 3.2.1) to the current position of the UAV. At this point, the UAVs were prepared and the experiments were performed. All of the real-world experiments were performed using the planar response functions (Equation (2.9)).

5.6.1 Description of experiments

Firstly, the algorithm was tested with two UAVs.[3] After the startup procedure, target for both of them was set to a single point to test the collision avoidance properties of the algorithm. The collision avoidance was further tested by setting target of one of the UAVs to the exact position of the other one. During these tests, the agents stayed in a safe distance away and thus we continued to the shared goal scenario, in which the targets of both of the agents were set to the same point further away from them. The controller performed as expected during the movement, maintaining a safe distance between the agents. Lastly, a leader follower scenario was tested, in which, the target controller of one of the two UAVs was disabled completely. The follower was able to stay near the leader even though it had to push against the wind to stay close. Afterwards a few changes of the leader's target were tested and the follower was mostly able to keep similar distance to the leader during the movement.

On the second attempt, the same scenario was tested with 5 UAVs. The initial setting of same target as well as the first change of target for all agents worked as expected. However, after the swarm arrived to its second target, some oscillations were induced and the experiment was stopped for safety reasons. For future attempts, the drag coefficient D was increased to lower the chances of inducing oscillations during flight. During later inspection, it was found that there were some issues with the Wi-Fi connection used by the FADAR system, which were likely the source of the oscillations, because the system was getting outdated inputs.

[3]This experiment was not recorded as its primary objective was to evaluate the other systems (such as FADAR).



(a) Trajectories from 95 to 197 s.

(b) Trajectories from 197 to 251 s.

Figure 5.7: Parts of the trajectories during the shared goal part of the third real-world experiment. Markers on trajectories are spaced in 5 s intervals, gray lines connect the same times.

On the third attempt, only 3 UAVs were available. There were no issues with oscillations, and the system was tested with a scenario similar to the first real-world test. Longer flights between targets were also tested both with all agents having the target, as well as one of them being a follower and only two having the target. The rest of this section is dedicated to analyzing parts of this experiment in more detail.

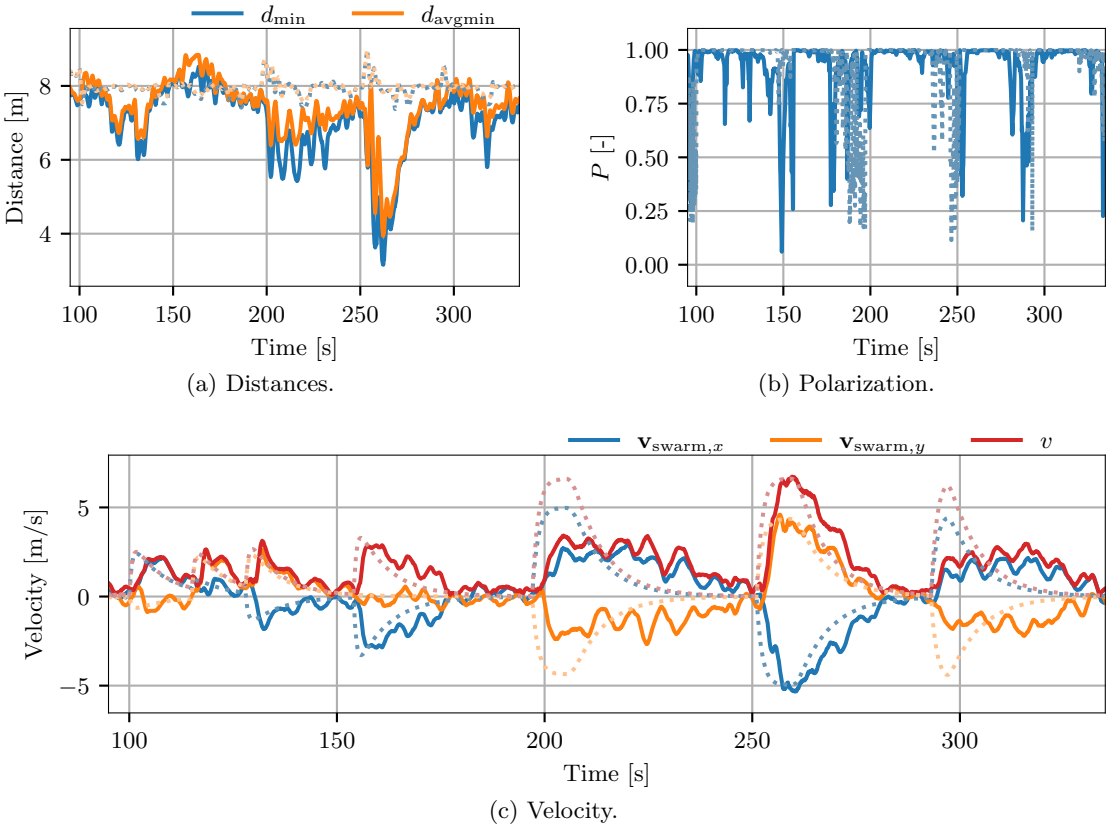
5.6.2 Shared goal

Firstly, we will focus on the part with shared goal known to all agents. At first (between around 100–200 s) there were multiple changes of the target, each moving the reference ca. 30 m from the previous one. After finishing these shorter flights, at around 200 s, the UAVs were given a target that was more than 100 m away. These parts of the trajectories are shown in Figure 5.7. After reaching the target (just after 250 s), they ware sent back in the opposite direction and lastly (around 290 s) the target was set to a position, which was around 70 m away.

For comparison, the same experiment was also performed in the custom simulator by giving the agents the same targets as in the real-world experiment. The behavior observed in the real-world matched the simulated results. Figure 5.8 shows the comparison of the real-world and simulated metrics. Comparing these results, we can clearly see, that even though the simulation is greatly simplified, the simulated results for polarization match the real-world results well during the travel (Figure 5.8b). During the periods of slowing down near target, the real-world data shows lower alignment compared to the simulations, which is likely to be caused by the noisier environment and outside disturbances of the movement[4].

Comparing the velocities, even though the experimentally measured data is much noisier, we see similar patterns in both the simulated and real-world data (Figure 5.8c). Probably the most noticeable mismatch between the two is the difference in speed measured between

[4]Some of the differences can also be attributed to slight differences in timing of the changes of targets and the flight speed of the UAVs, which was different between simulation and reality.



(a) Distances.

(b) Polarization.

(c) Velocity.

Figure 5.8: Comparison of metrics for the real-world experiment and simulation. Full line is real-world data, dotted line is simulated data.

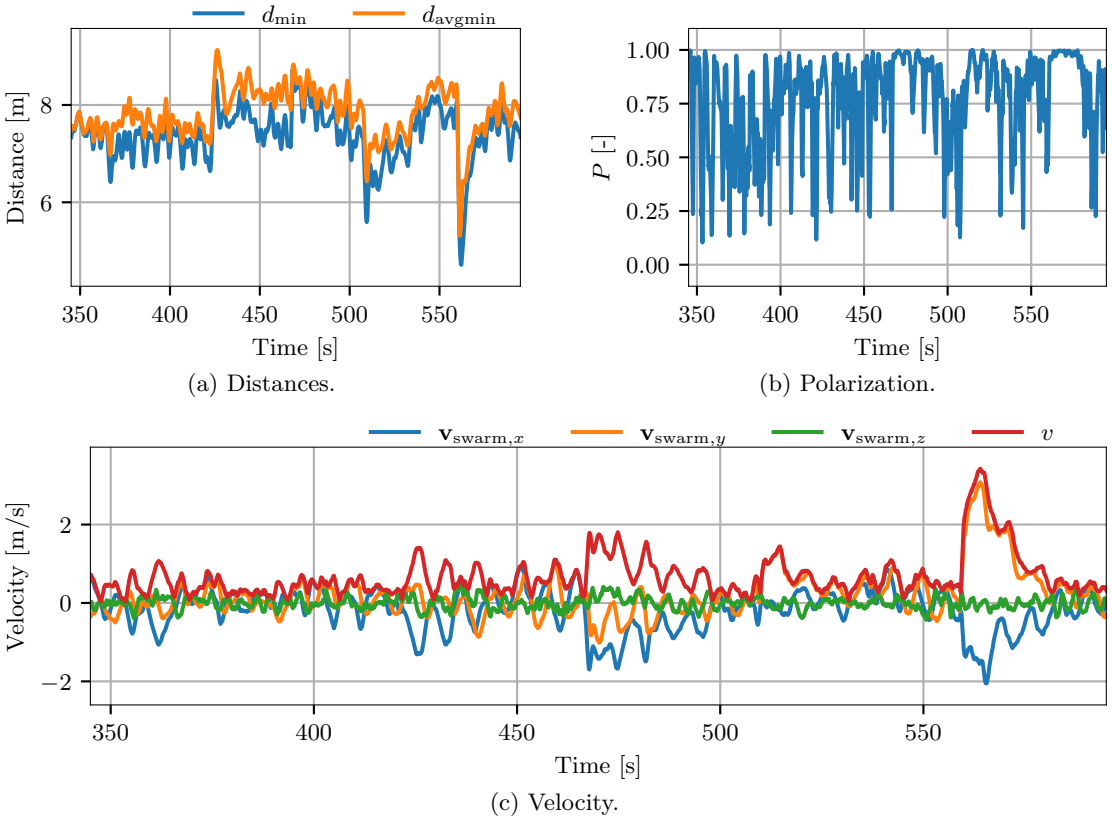
the first and second long flight, which was not present in the simulated data. This discrepancy was caused by wind during the real-world experiment. The first long flight (ca. 200–250 s) was going almost directly against the wind, causing the UAVs to fly slower. Afterward, the return flight (ca. 250–280 s) was in the direction of the wind, helping the UAVs to fly faster.

Lastly, when comparing the mutual distances (Figure 5.8a), we see that in both cases, the average and minimal distances stay around or slightly below 8 m. In the case of the real-world experiment, there was one large drop in minimal distance around 260 s. This was at the start of the long return flight and was caused by a delayed request to change the target of the controller. This delayed agent was the one closest to the target, meaning the other two agents started flying towards it. As they started approaching closer, the swarming controller started moving the delayed agent while also slowing down and redirecting the other two agents, preventing violation of the 3 m safety area around the UAVs and a potential collision.

The comparison with simulation shows that the resulting behaviors in the custom simulator are accurate enough to be replicable in the real world, further supporting the results of the simulations with other scenarios that were not tested during the real-world experiments.

5.6.3 Leader–follower

In the second part of this experiment, the target controller of one of the UAVs was disabled to test the leader–follower behavior. After turning of the target, we first kept the



(a) Distances.

(b) Polarization.

(c) Velocity.

Figure 5.9: Metrics measured during the leader–follower part of the real-world experiment.

other UAVs in place. This part showed that the swarm controller was able to keep the follower close to them even in the wind[5]. Afterwards, a few changes to the target of the leaders were done to check, how well it would be able to follow. The metrics for this part of the experiment are plotted in Figure 5.9.

Analyzing the distances (Figure 5.9a), we can see that similarly to the first part of the experiment, the average minimal distance is around 8 m. However, compared to the part where all agents had the target controller (Figure 5.8a), we see a slightly larger difference between the minimal and average minimal distance. Because of wind during the experiment, the follower agent was further from the two leaders as it only used the visual swarm controller to control its position, which is not designed to counteract outside influences.

After disabling the target controller of one of the agents (at around 351 s) there was almost no movement. The follower first rotated around the two other UAVs and then drifted a little further because of the wind. After that, the swarm was mostly stable in place with a close to zero velocity (Figure 5.9c). Since there was almost no movement of the whole swarm, the polarization metric for this part is mostly irrelevant.

Furthermore, we proceeded to test the ability of the swarm controller to follow the leaders. At around 420 s, we started issuing commands to the leaders to go to specified positions. Figure 5.10 shows the trajectories of all three UAVs during this part of the experiment (420–595 s). From there, we can clearly see that the follower was able to stay close to the leaders throughout the whole experiment.

[5]Since the target controller of the follower was disabled, the only position control was done by the swarming controller.

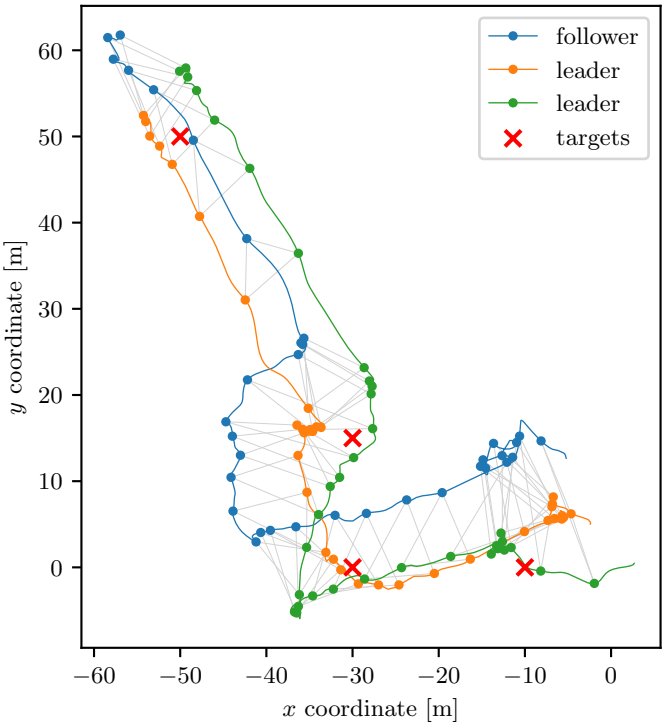


Figure 5.10: Trajectories of all UAVs during part of the leader–follower experiment (420–595 s). Beginning of the trajectories is near $[0, 0]^{\mathrm{T}}$. Markers on trajectories are spaced in 5 s intervals, gray lines connect the same times.

Figure 5.9b shows the polarization during this experiment. During the short changes in target (around 420–460 s) the polarization oscillates and behaves similarly to when there was no movement at all. This is likely caused by the low overall speed of the swarm (Figure 5.9c) during this part. When the distances of between targets increased, the speed was higher and subsequently, the polarization grew as well. We see that during the flight to further away targets (around 460–500 s, 510–540 s and 560–585 s), the swarm has high polarization, thanks to both higher speed and having more time to align the movement. The higher polarization is not as stable as in the first part of the experiment with a shared goal, however, here, the alignment of the follower is influenced only using the swarm controller.

In summary the visual swarming algorithm was able to perform the real-world experiment as expected. Both the shared goal and leader–follower scenarios resulted in similar behaviors as in the simulations. In the leader–follower scenario, the controller was also able to counteract the impacts of external disturbances, such as wind, without using any position information on the follower.

6 Conclusion

In this work, a method for extending a 2D visual swarming algorithm [12], [13] to work with UAVs in 3D environment was proposed. This included updating the observation format, the inner workings of the controller and changing the outputs to match the capabilities of multirotor UAVs.

The proposed algorithm was successfully implemented in C++ with interfaces for both a custom developed lightweight simulator and the MRS UAV System [14]. The implemented controller was evaluated in three environments with increasing fidelity: the custom simulator to test the desired behaviors of the swarm, the MRS Multirotor Simulator to evaluate the same behavior with more complex dynamical model and in the real world to prove the validity of the simulated results.

The simulations showcased that the controller performs well in different scenarios, ranging from globally known shared goal, through leader–follower scenarios with only limited knowledge of targets, to emergent behaviors such as flocking in a direction that resulted from a consensus of the agents. In all of these scenarios, the visual swarming algorithm was able to keep the swarm coherent, align the movement of all UAVs and maintain safe distances between the agents.

Lastly, the system was also successfully tested in real-world experiments with the shared goal and leader–follower scenarios, showing behavior similar to what was observed during simulations. Through a comparison with a simulation of the same experiment, we have shown that the real-world measured results are consistent with the simulations. This comparison also further supports the results observed during the experiments in simulators.

6.1 Future work

This work showed the potential of the purely visual swarming controller for use on UAVs, however, there are still areas to be explored. One of these areas is obtaining the input observation. Creation of a system for obtaining the observations without knowing positions of the others directly from some onboard sensors (e.g., camera, LiDAR, etc.) would allow for real-world evaluation of the entire swarming system with no communication required. This could be achieved by adapting existing approaches to produce the required observation format. The approaches that could be adapted include machine vision models (e.g., convolutional neural networks, vision transformers) and relative localization systems (e.g., UVDAR). An interesting approach would be combining the model with a bio-inspired neighbor density estimator proposed by Křížek et al. [21].

Another potential extension would be adding more classes to the input semantic map to allow more complex behaviors. By setting different response functions (and thus different behavior rules) for each class, the controller would be able to keep the swarm coherent, while simultaneously avoiding collisions with environment and maintaining a safe distance from other UAVs that are not part of the swarm.

7 References

[1] L. Marconi et al., "The SHERPA project: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in alpine environments," in *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2012, pp. 1–4. DOI: 10.1109/SSRR.2012.6523905.

[2] T. A. Vidal-Calleja, C. Berger, J. Solà, and S. Lacroix, "Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain," *Robotics and Autonomous Systems*, vol. 59, no. 9, pp. 654–674, 2011, ISSN: 0921-8890. DOI: 10.1016/j.robot.2011.05.008.

[3] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of Multiple UAVs for Persistent Surveillance: Algorithm and Flight Test Results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, 2012. DOI: 10.1109/TCST.2011.2167331.

[4] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87, New York, NY, USA: Association for Computing Machinery, 1987, pp. 25–34, ISBN: 0897912276. DOI: 10.1145/37401.37406.

[5] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel Type of Phase Transition in a System of Self-Driven Particles," *Physical Review Letters*, vol. 75, pp. 1226–1229, 6 1995. DOI: 10.1103/PhysRevLett.75.1226.

[6] H. Tanner, A. Jadbabaie, and G. Pappas, "Stable flocking of mobile agents, part I: fixed topology," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 2, 2003, 2010–2015 Vol.2. DOI: 10.1109/CDC.2003.1272910.

[7] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: algorithms and theory," *IEEE Transactions on Automatic Control*, vol. 51, no. 3, pp. 401–420, 2006. DOI: 10.1109/TAC.2005.864190.

[8] M. Saska, J. Vakula, and L. Přeučil, "Swarms of micro aerial vehicles stabilized under a visual relative localization," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3570–3575. DOI: 10.1109/ICRA.2014.6907374.

[9] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982. DOI: 10.1109/TIT.1982.1056489.

[10] P. Dames, "Distributed multi-target search and tracking using the PHD filter," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2017, pp. 1–8. DOI: 10.1109/MRS.2017.8250924.

[11] M. Boldrer, L. Palopoli, and D. Fontanelli, "A unified Lloyd-based framework for multi-agent collective behaviours," *Robotics and Autonomous Systems*, vol. 156, p. 104 207, 2022, ISSN: 0921-8890. DOI: 10.1016/j.robot.2022.104207.

[12] R. Bastien and P. Romanczuk, "A model of collective behavior based purely on vision," *Science Advances*, vol. 6, no. 6, eaay0792, 2020. DOI: 10.1126/sciadv.aay0792.

[13] D. Mezey et al., "Purely vision-based collective movement of robots," 2024. DOI: 10.48550/arXiv.2406.17106.

[14] T. Baca et al., "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, 1 2021. DOI: 10.1007/s10846-021-01383-5.

[15] *Robotic Operating System*, version ROS Noetic Ninjemys. Accessed: Jan. 8, 2025. [Online]. Available: https://www.ros.org.

[16] T. Baca, *MRS Multirotor Simulator*, 2024. Accessed: Jan. 8, 2025. [Online]. Available: https://github.com/ctu-mrs/mrs_multirotor_simulator.

[17] V. Walter, M. Saska, and A. Franchi, "Fast Mutual Relative Localization of UAVs using Ultraviolet LED Markers," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2018, pp. 1217–1226. DOI: 10.1109/ICUAS.2018.8453331.

[18] V. Walter, N. Staub, A. Franchi, and M. Saska, "UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2637–2644, 2019, ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2901683.

[19] *nimbro_network - ROS transport for high-latency, low-quality networks*. Accessed: Apr. 10, 2025. [Online]. Available: https://github.com/AIS-Bonn/nimbro_network.

[20] V. Walter, *UVDAR Multirobot Simulator*, 2024. Accessed: Jan. 8, 2025. [Online]. Available: https://github.com/ctu-mrs/uvdar_multirobot_simulator.

[21] M. Křížek, M. Vrba, A. B. Kulaš, S. Bogdan, and M. Saska, "Bio-inspired visual relative localization for large swarms of UAVs," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 11 825–11 831. DOI: 10.1109/ICRA57147.2024.10610100.

A Parameters

Experiment	S_{rep}	S_{att}	S_{vis}	D	K^*
custom simulator (except emergent flocking)	80	2.7	1.5	0.05	-
custom simulator – emergent flocking	80	2.7	1.5	0.05	0.5
real-world experiment (Sections 5.6.2 and 5.6.3)	80	3	10	0.55	-
simulation of the real world experiment	80	3	10	0.75**	-

* If turning limiter was used, otherwise empty.
** Higher than in the real world experiment to compensate for the physical drag of the real UAVs.

Table A.1: Swarming controller parameters used during the experiments.

Controller	K_{p}	K_{i}	K_{d}	M
custom simulator – target controller	0.2	0.0	0.9	0.5
custom simulator – speed controller*	0.9	0.1	0.5	0.5
real world – altitude band	0.7	0.05	1.3	5
real world – target xy component	0.2	0.005	0.9	4
real world – target z component	0.7	0.05	1.3	5

* Used for the emergent behaviors experiments. Target speed was 2 m/s.
$K_{\text{p}}, K_{\text{i}}, K_{\text{d}}$ are gains of the controller.
M is the maximal allowed norm of the output
(if the output was higher, it was clipped).

Table A.2: PID controllers parameters.

B Supplementary materials

- source code
  - `vision-sim3d` – the custom simulator together with the implementation of the visual controller to interface with it
  - `swarm-simulator-visualization` – visualizer for simulation results from the custom simulator
  - `ros` – ROS packages with implementation used in the real-world experiments and MRS Multirotor Simulator
- video recordings of the real-world experiment
  - `shared_1.mp4` – first part of the shared goal experiment
  - `shared_2.mp4` – second part of the shared goal experiment
  - `lf_1.mp4` – first part of the leader–follower experiment
  - `lf_2.mp4` – second part of the leader–follower experiment
- video recordings of some of the simulations
  - `sim_shared.mp4` – shared goal scenario (analyzed in Section 5.2.2)
  - `sim_emergent_x4.mp4` – emergent behavior scenario with constrained environment and limited turning rate [this recording is sped up 4 times] (analyzed in Section 5.2.4)

C Use of AI

During the preparation of this thesis, the author used OpenAI ChatGPT[1]. This AI tool was used exclusively to help create data processing scripts for extracting the required data from telemetry recordings created during the real-world experiments. All of the data processing was done by the generated code, not by the AI. The author reviewed all of the generated code and takes full responsibility for it.

[1]Available at: https://chatgpt.com/.