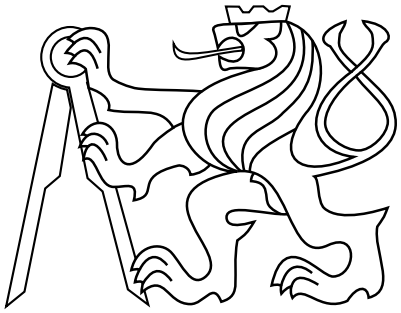
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Flocking of Unmanned Aerial Vehicles Based on a Higher-Order Boids

Bachelor's Thesis

Daniel Švehla

Prague, April 2025

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

Supervisor: Ing. Martin Jiroušek

Acknowledgments

Firstly, I would like to express my gratitude to my family for their unwavering support throughout my whole life. I would also like to extend a special thanks to my supervisor, Ing. Martin Jiroušek, whose guidance and constructive feedback were essential to the completion of this thesis. I am also grateful to Multi-robot Systems Group (MRS) as a whole and its members for providing access to the hardware and and their assitance in making the real-world experiment a reality. Finally, I would like to thank all my friends for their support during my studies.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Švehla Daniel** Personal ID number: **516424**

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 a Higher-Order Boids

Bachelor’s thesis title in Czech:

Hejno bezpilotních letounů založené na Boids modelu vyššího řádu

Guidelines:

The objective of this thesis is to design, implement, and validate an improved, more agile Boids-based algorithm for swarm control of multirotor unmanned aerial vehicles (UAVs). The proposed extension should incorporate the second and higher derivatives of the UAV position to enhance swarm agility and responsiveness.
Student tasks:
1) Familiarize yourself with the topic of multi-robot swarm systems, with a particular focus on studying article [1] and the practical implementation of the Boids algorithm as described in [3].
2) Understand the ROS framework, MRS UAV system [4] and integrate the model in decentralized fashion.
3) Implement an enhanced Boids model that utilizes the second and higher derivatives of agents' positions in C/C++ language.
4) Evaluate the performance of the enhanced Boids model on large swarms and compare it with the baseline Boids model.
5) Analyze the robustness of the proposed model under imperfect data conditions.
6) Conduct real-world experiments to validate the algorithm, if hardware availability permits.

Bibliography / sources:

[1] Reynolds, C. W. (1987). "Flocks, Herds, and Schools: A Distributed Behavioral Model." Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87). ACM, 25-34
[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] Pavel Petráček, Viktor Walter, Tomáš Báča and Martin Saska. Bio-Inspired Compact Swarms of Unmanned Aerial Vehicles without Communication and External Localization. Bioinspiration & Biomimetics 16(2):026009, December 2020
[4] 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
[5] 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.

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): Švehla Daniel
Personal number: 516424
Programme name: Open Informatics

declare that I have elaborated the bachelor's thesis entitled

Flocking of Unmanned Aerial Vehicles Based on a Higher-Order Boids

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 15.05.2025

Daniel Švehla
.....
student's signature

Abstract

This thesis presents a new, fully-decentralized swarming algorithm for Unmanned Aerial Vehicles (UAVs) that addresses a critical limitation of classical boids flocking models: dependence on relative velocity measurements, which are challenging to measure reliably in decentralized systems. By integrating higher-order derivatives - specifically, relative acceleration and orientation of neighbouring agents - the proposed method eliminates the need for noisy velocity measurements. The algorithm and its resilience against severe sensor noise is rigorously evaluated in simulations. Further, real-world outdoor flight experiments are conducted to validate real-world performance. Results confirm that the algorithm maintains safe separation, cohesion and alignment even under severe sensor noise, outperforming traditional boids implementations in practical scenarios.

Keywords Unmanned Aerial Vehicles, UAV, Drones, Swarm, Flock, Boids, Decentralized, Bio-Inspired, Robotics, Automatic Control

Abstrakt

Tato práce představuje nový, plně decentralizovaný rojový algoritmus pro bezpilotní prostředky (UAV), jenž řeší zásadní omezení klasických modelů hejn typu „boids“: závislost na měření relativních rychlostí, která jsou v decentralizovaných systémech obtížně spolehlivě zjistitelná. Integrací stavových informací vyššího řádu - konkrétně relativního zrychlení a orientace okolních agentů - navrhovaná metoda eliminuje potřebu zašumělých měření rychlosti. Algoritmus a jeho odolnost vůči výraznému senzorickému šumu jsou podrobně vyhodnoceny v simulacích. Navíc byly provedeny venkovní letové experimenty v reálném prostředí, které potvrdily praktickou použitelnost metody. Výsledky prokazují, že algoritmus udržuje bezpečnou separaci, soudržnost i zarovnání i za podmínek silného senzorického šumu a v praktických scénářích překonává tradiční implementace boids.

Klíčová slova Bezpilotní Prostředky, UAV, Drony, Roje, Hejna, Boids, Decentralizace, Přírodou-Inspirované, Robotika, Autonomní Řízení

Abbreviations

CTU Czech Technical University

FOV Field of View

GNSS Global Navigation Satellite System

GPS Global Positioning System

MPC Model Predictive Control

MRS Multi-robot Systems Group

ROS Robot Operating System

RTK Real-time Kinematics

UAV Unmanned Aerial Vehicle

UV UltraViolet

UWB Ultra WideBand

UVDAR UltraViolet Direction And Ranging

Contents

1 Introduction

Unmanned aerial vehicles (UAVs), commonly known as drones, have evolved remarkably over the last two decades. From niche hobbyist platforms to sophisticated systems with uses across environmental exploration [10], disaster response [16], infrastructure inspection [11], and even planetary exploration [7]. As individual UAV capabilities continue to improve, research focus has increasingly shifted toward coordinating multiple UAVs operating collaboratively as a swarm.

Collective behaviour (Fig. 1.1) is one of nature's most successful survival strategies: flocks of birds, schools of fish, and swarms of insects routinely achieve tasks - migration, predator evasion, foraging - that would overwhelm a solitary animal. Translating these bio-inspired principles to robotics holds the promise of systems that are **scalable** and **robust**. Swarm robotics offers the potential to accomplish complex tasks more efficiently, robustly, and scalably than single vehicles ever could.

Decentralized swarms combine the scalability of local, bio-inspired control laws with the fault-tolerance of fully distributed decision-making. Centralized architectures funnel sensing and control through a hub that can be jammed, overloaded, or physically destroyed. In contrast, decentralized systems keep all control loops on-board, so the swarm continues to function even when individual UAV or communication links go down. Because each agent only needs relative neighbour states that can measured locally, the decentralized approach eliminates both the bandwidth bottlenecks and the vulnerability of a master node.

Figure 1.1: Example of bird flocking in nature.

1.1 Related work

Swarming has been field of interest for many decades. Even today, many swarming algorithms are build upon flocking rules introduced decades ago by fundamental swarming research. Craig Reynolds' 1987 Boids paper [24] introduced one of the most influential swarming model, which is especially relevant for this thesis. Boids model consists of three flocking rules (separation, alignment, cohesion) and each agent in the swarm follows these rules to produce lifelike flocking behaviour. Another fundamental model is Vicsek model [23] introduced in 1995, in which all agents move at constant speed while trying to align with its neighbours.

The main issue with Boids, Vicsek and other similar models is that they consider each agent as dimensionless and massless particle, which usually results in non-optimal performance in robotic applications without further adjustments. The primary focus and motivation behind these models were non-robotic applications, where these characteristics were not problematic. Such applications include computer graphics, where models can be used to simulate large amount of particles [21], or computer games, in which the movement of in-game characters can be computed [22].

Advances in UAV hardware resulted in a lot more interest in aerial swarming. Many aerial swarming models based on Boids were proposed and validated in real-world deployments [14, 18, 19], including deployments with onboard relative localization [17]. Swarming has been extensively researched in MRS for many years with applications in various fields [1, 2, 3, 4, 6]. Furthermore, MRS developed and deployed Boids swarming model with GPS localization [15]. Similar models were later deployed in fully-decentralized fashion utilizing vision-based relative localization [9, 12].

Accurate and robust relative localization has often been a barrier when deploying aerial swarms. The easiest solution is to use GNSS, but that requires direct communication between agents and in practice is not very accurate without the use of RTK. GNSS signal can also be easily lost or even jammed, reducing the overall robustness of the swarm. Vision-based relative localization methods can be more accurate and robust making them ideal for aerial swarms. UVDAR [13] is one such system developed by MRS. These methods however present their own issue - they cannot directly measure velocity data. While the velocity can still be computed by differentiating multiple positions, it often introduces noise.

1.2 Contributions

In this thesis we present a new, fully-decentralized swarming algorithm for UAVs that builds upon the classic boids flocking rules by incorporating higher-order state information, so that each agent reacts to relative acceleration and orientation of its neighbours instead of their relative velocities, which are difficult to measure in decentralized fashion. The algorithm is validated in simulation with swarms ranging from a few up to a hundred of UAVs, and then on a hardware platform in real-world outdoor flights utilizing robust Kalman-filter state fusion. The results show that the proposed method preserves collision-avoidance and alignment under severe sensor noise, thereby advancing the state of the art in aerial swarming better suited for real-world deployment.

1.3 Problem Definition

We define **swarm** as group of at least 3 autonomous and decentralized UAVs. In this thesis, terms **flock** and **swarm** are used interchangeably. Main goal is to develop new fully-decentralized swarming model specifically designed for UAVs and their dynamics by integrating higher-order derivatives of positions, specifically orientation. Decentralization is key part of the problem, fully-decentralized swarming model cannot communicate across UAVs and must run completely independently on each UAV.

For measuring positions, we aim to use UVDAR 2 system. Orientation measurement system is simulated by sending orientation data, since such systems are not yet ready. The swarming model itself remains fully decentralized, meaning that it is not aware of this communication.

The swarming model receives positions and orientations from these mentioned systems and outputs velocity reference vector, which is routed to the onboard MRS MPC flight controller. The swarming model is coded fully in C++ and integrated within Robot Operating System (ROS) and MRS UAV System, allowing for real-world deployments.

1.4 Mathematical notation

Below is a Table 1.1 defining mathematical notation used throughout this thesis. This notation holds true, unless specified otherwise.

x	scalar
$\mathbf{x}, \mathbf{q}$	vector, unit quaternion
$\hat{\mathbf{x}}$	normalized vector
$\|\mathbf{x}\|$	norm of vector
$\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$	elements of the standard basis
$\mathbf{X}$	matrix
$\mathbf{I}$	identity matrix
$\mathbf{R}_{\mathcal{A}}^{\mathcal{B}}$	rotation matrix representing transformation from frame $\mathcal{A}$ to frame $\mathcal{B}$
$\mathbf{R}_a(b)$	rotation around a axis by b angle
$\mathbf{x}^\intercal$	transposed vector
$\mathbf{X}^\intercal$	transposed matrix
$\mathbf{qxq}^{-1}$	vector $\mathbf{x}$ rotated with unit quaternion $\mathbf{q}$, $\mathbf{x}$ treated as pure quaternion
μ	mean
σ	standard deviation

Table 1.1: Mathematical notation, nomenclature and notable symbols.

2 Swarming Algorithm

The original boids algorithm introduced three fundamental rules - separation, cohesion and alignment. In this chapter, we present a new swarming algorithm that implements the separation and cohesion behaviours but replaces alignment with a novel tilt-matching mechanism.

2.1 Reference Frames

A reference frame (also known as a coordinate frame) is a defined coordinate system used to precisely describe and interpret the position, orientation, and motion of the UAV (Fig. 2.1). Selecting an appropriate reference frame is critical for navigation and control. Below are definitions of reference frames used throughout this thesis.

- **world** - Right-handed rectangular Cartesian coordinate frame fixed to the ground at defined point with z axis pointing above the ground.
- **body** - Body frame of the UAV, z-axis is parallel to the thrust force produced by the propellers.
- **local** - xyz axis are parallel to those of world frame, origin at the centre of the UAV. Useful for describing positions, velocities and accelerations relative to the UAV.

In this thesis, we assume that all UAVs have the same world-frame.

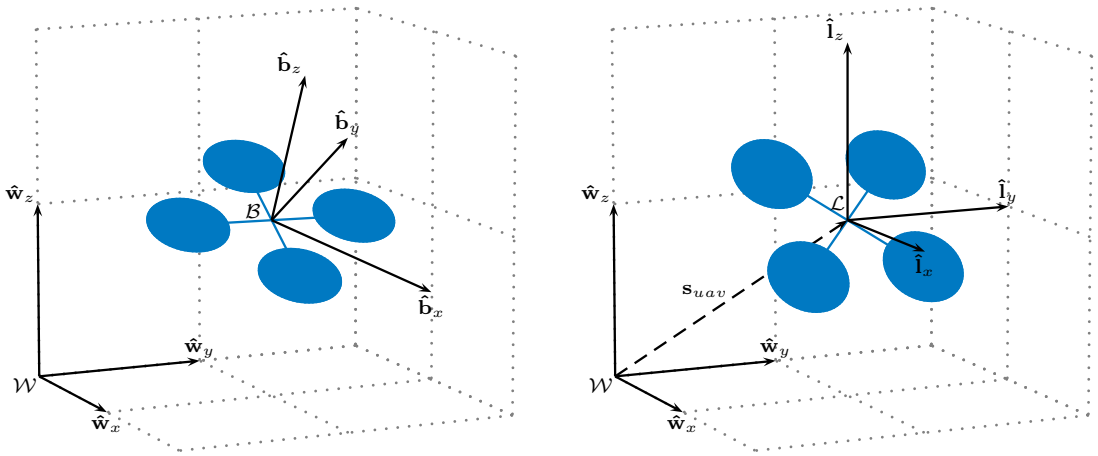


Figure 2.1: Visualisation of reference frames $\mathcal{W}$ (world), $\mathcal{B}$ (body) and $\mathcal{L}$ (local).

2.2 Algorithm

Before turning to swarming model details, following two terms are defined to classify UAVs:

- **Observer** - Every UAV running the algorithm.
- **Target** - Every UAV visible to given observer.

Furthermore, the target state vector $\mathbf{s}$ is defined as follows:

$$\mathbf{s} = \left[\mathbf{p}^\intercal, \mathbf{q}^\intercal\right]^\intercal,$$

$$\mathbf{p} = \left[p_x, p_y, p_z\right]^\intercal,$$

$$\mathbf{q} = \left[q_w, q_x, q_y, q_z\right]^\intercal,$$

where $\mathbf{p}$ is target position in observer-local-frame and $\mathbf{q}$ is target orientation in observer-local-frame in form of unit quaternion. Every observer keeps track of all visible targets, their states and its own orientation $\mathbf{q}_o$ in observer-local-frame.

Note that the observer does not know its position and that target state does not include velocity, which makes this algorithm ideal for real-world deployment in decentralized fashion and is one of the main reasons, why alignment (velocity) matching is replaced by tilt matching. We will go into more detail in later chapters.

Separation

The separation rule enforces safe distance between the observer and targets in order to avoid collisions. At first, separation rate for every target in radius r is calculated by separation rate function. In its simplest form, it can be represented as a linear function of the distance. However, more complex function is needed to produce smoother and more realistic avoidance behaviours. For that reason, following separation rate function (Fig. 2.2) is proposed:

$$g(d) = \frac{r - d}{z\,(d - d_{safe})},\ d \in (d_{safe}, r]$$

where d is distance, r is separation radius, z is decay rate and d_{safe} is minimum safe distance. If target distance d is equal or lower than d_{safe}, it is clamped to $d_{safe} + 0.1$.

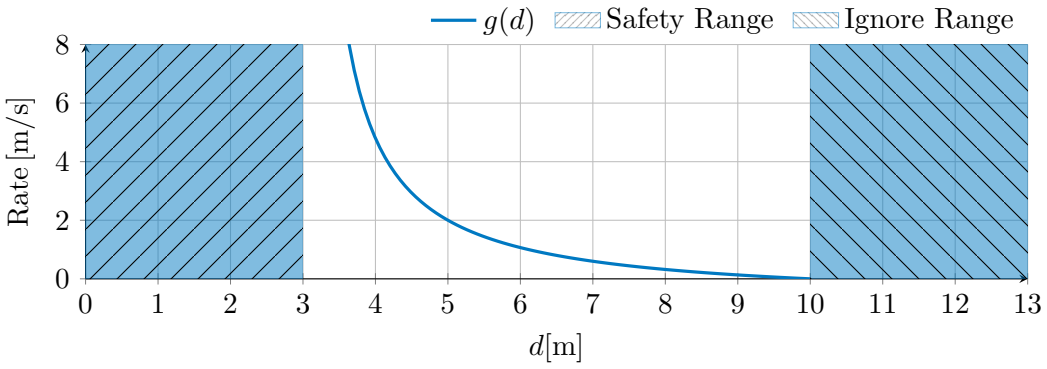


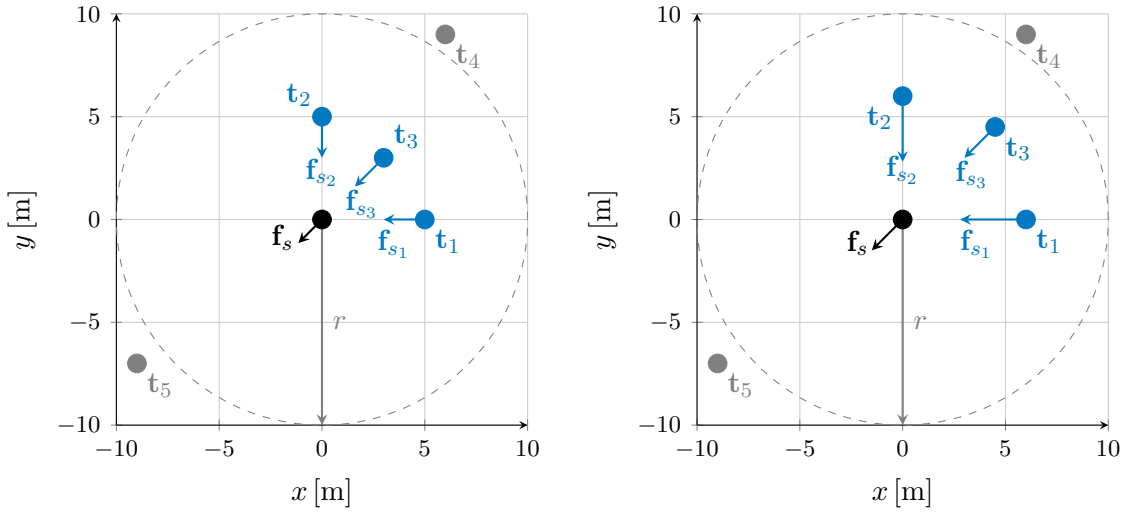
Figure 2.2: Separation rate, $r = 10$, $z = 1.25$, $d_{safe} = 3$.

The separation vector for each target t in separation radius r is then calculated as

$$\mathbf{f}_{s_t} = -\hat{\mathbf{p}}_t \; g(\|\mathbf{p}_t\|),$$

The overall separation vector is obtained by averaging these individual vectors (Fig. 2.3):

$$\mathbf{f}_s = \frac{\sum_{t=1}^{N} \mathbf{f}_{s_t}}{N}.$$



(a) $r = 10$, $z = 1.25$, $d = 3$ (b) $r = 10$, $z = 1.25$, $d = 5$

Figure 2.3: Visual example of separation rule. Observer in the origin. Only blue targets $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$ within the separation radius r have impact on the final separation vector $\mathbf{f}_s$.

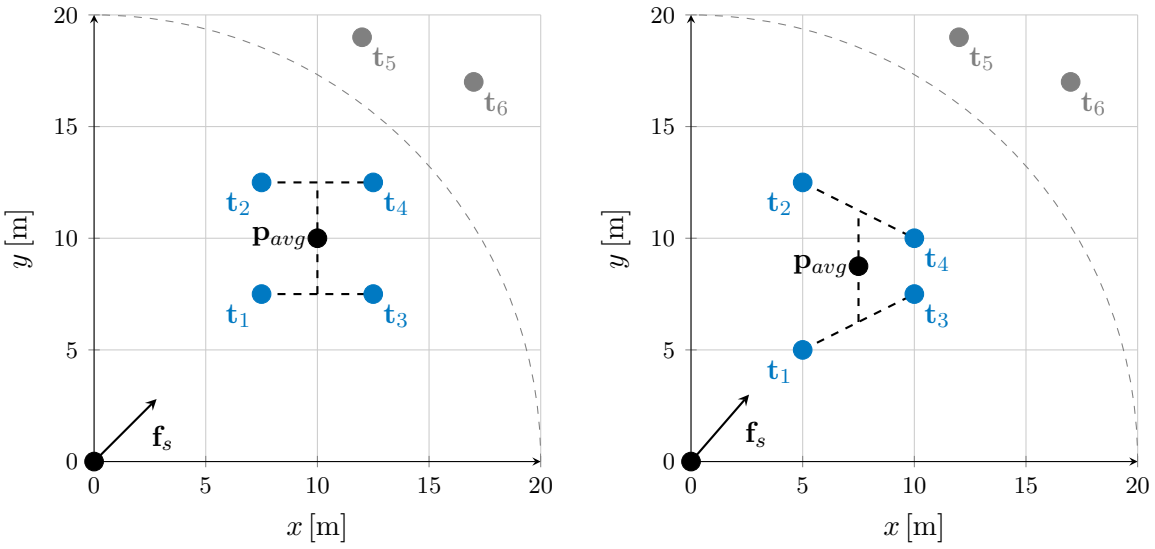
Cohesion

Opposite to separation rule, cohesion rule pulls the observer towards targets ensuring that the swarm maintains unity (Fig. 2.4). The cohesion vector is calculated by averaging positions of all visible targets and normalizing it:

$$\mathbf{p}_{avg} = \frac{\sum_{t=1}^{N} \mathbf{p}_t}{N},$$

$$\mathbf{f}_c = \hat{\mathbf{p}}_{avg}.$$

Normalizing the cohesion vector is an intentional design decision. While leaving it unnormalized might seem to improve responsiveness when chasing targets, doing so risks instability. As targets come into and out of visibility range, the vector's magnitude could fluctuate dramatically. By fixing its length, we ensure a consistent, stable response regardless of current swarm behaviour.



(a) Example 1.

(b) Example 2.

Figure 2.4: Visual example of cohesion rule. Observer in the origin. Only blue targets $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3$, $\mathbf{t}_4$ within visibility range are processed. Cohesion vector $\mathbf{f}_c$ multiplied by 4 for better readability.

Tilt Alignment

In the traditional boids model, alignment of the swarm is maintained by matching velocity of targets. This requires accurately measuring relative velocity, which-without direct communication-demands tracking multiple position samples and differentiating them numerically, a process that often introduces large noise in real-world UAV deployments. In contrast, a UAV's orientation can be measured directly and without delay.

Moreover, Reynolds' original formulation assumes point-mass model and just velocity matching fails to capture the complex UAV dynamics. For example, when a UAV suddenly reverses direction, its velocity vector may take several seconds to catch up, whereas its orientation changes instantaneously, making orientation-based alignment both more reliable and better suited to UAVs.

To calculate the alignment vector, we first need to calculate pseudo-acceleration vector to which the observer aligns. This is done by interpreting the z axis of the body frame in the world coordinate frame and then projecting the resulting pseudo-acceleration vector onto xy plane:

$$\mathbf{a}^+ = \mathbf{q}\hat{\mathbf{e}}_z\mathbf{q}^{-1},$$
$$\mathbf{a} = \left[a_x^+, a_y^+, 0\right]^\intercal.$$

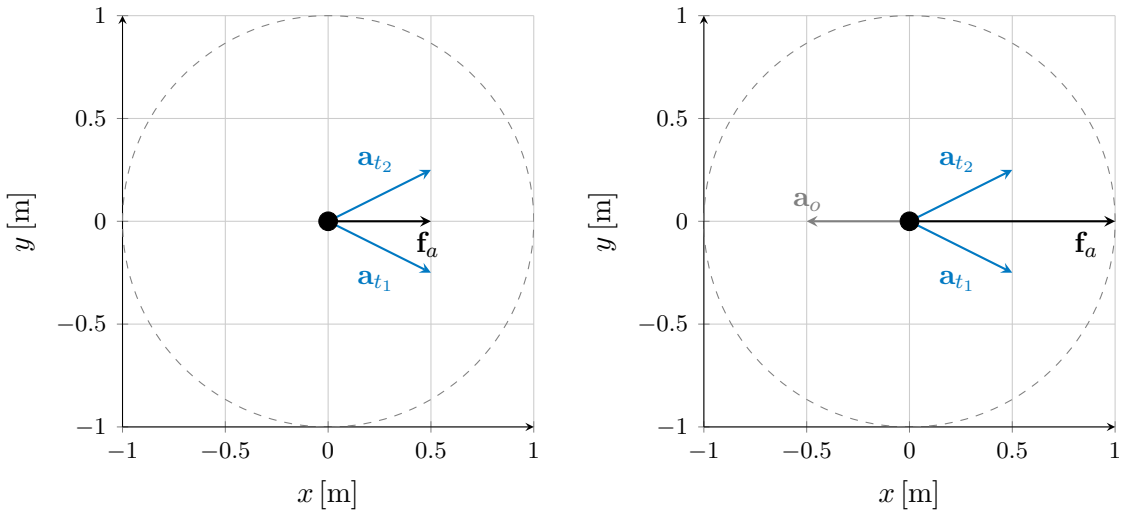
Next, the average target pseudo-acceleration is computed as

$$\mathbf{a}_{avg} = \frac{\sum_{t=1}^{N} \mathbf{a}_t}{N}.$$

Finally, the alignment vector (Fig. 2.5) is simply the difference between the average pseudo-acceleration of targets and observer's own pseudo-acceleration:

$$\mathbf{f}_a = \mathbf{a}_{avg} - \mathbf{a}_o.$$

Since the pseudo-acceleration always has zero z element, this approach is best suited for UAVs flying at relatively constant altitude. However, as will be shown in the next chapters, this approach works well even for large swarms with complex 3D formations.



(a) Zero observer pseudo-acceleration.

(b) Non-zero observer pseudo-acceleration.

Figure 2.5: Visual example of tilt alignment rule. xy plane onto which are the pseudo-accelerations projected. The circle represents the possible range of pseudo-acceleration vectors, $\mathbf{v}$ on the edge of this circle would represent UAV flying 90-degree sideways.

Goal Tracking

So far all rules were designed to achieve cohesive, well aligned swarm with safe UAV separation. Without any additional rules, UAVs would just swarm around some point, or the whole swarm could even start drifting into random direction. One additional rule needs to be introduced, that will push the observer towards given goal. This rule is special in that it is not influenced by any target. The tracking vector is simply defined as:

$$\mathbf{f}_t = \hat{\mathbf{g}},$$

where $\mathbf{g}$ is goal position in observer-local-frame. Every observer can have its own separate goal and thus this tracking vector can differ for each individual observer in the swarm.

Combining Vectors

The complete swarming algorithm is a result of the 4 described vectors. These vectors are weighted and added resulting in the final velocity vector:

$$\mathbf{f} = w_s\mathbf{f}_s + w_c\mathbf{f}_c + w_a\mathbf{f}_a + w_t\mathbf{f}_t,$$

where w_s, w_c, w_a and w_t are weights. The resulting velocity vector $\mathbf{f}$ is in world-frame and represents the direction and velocity in which the observer shall fly. The onboard controller of the observer is then responsible for matching this velocity vector.

3 Estimation Pipeline

Making the algorithm robust against sensor noise is critical for real-world deployment, where the lack of precision of position and orientation measurements can significantly impact overall performance. In this chapter we introduce the deployment platform and its related on-board localization systems. Effective kalman filtering technique for position measurements is then proposed and validated.

3.1 Hardware Platform

RoboFly (Fig. 3.1) was chosen as target hardware platform for this thesis. RoboFly is lightweight research platform ideal for both indoor and outdoor testing developed by F4F[1] - a MRS spin-out. The platform runs upon the MRS UAV System and is equipped with Raspberry Pi 5 computer[2], enabling demanding workloads including computer vision algorithms. This is supported by two wide-angle cameras, one facing forwards and the other backwards. Furthermore, the new UVDAR 2 localization system is supported - main reason for choosing RoboFly platform.

Figure 3.1: RoboFly UAVs at Temešvár MRS camp.

3.2 Measuring Positions

Robust system for measuring target positions is often the biggest hurdle when trying to deploy swarming algorithms in truly decentralized fashion. UVDAR 2, the new and improved version of the well established UVDAR [13] developed in MRS, aims to provide reliable and very accurate positioning system in real-world outdoor conditions. This is achieved by using UV markers on every UAV in combination with UV-filtered cameras (Fig. 3.2), resulting in great resistance against the widely varying illumination conditions encountered in outdoors environments. UVDAR 2 uses UV markers just to measure bearing while the distance is measured by time-of-flight UWB.

[1] F4F details: https://fly4future.com/about-us/.
[2] Raspberry Pi 5: https://www.raspberrypi.com/products/raspberry-pi-5/

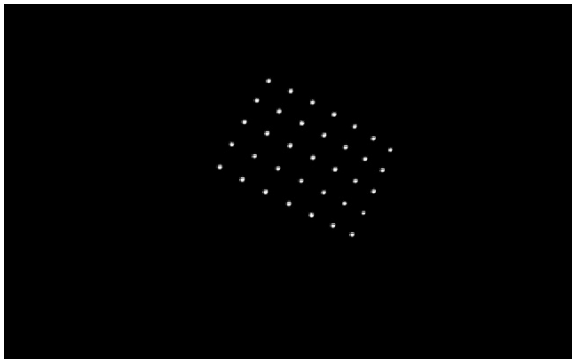


Figure 3.2: Example of what the UV-filtered camera sees. 640x400 camera resolution. Points represent UV markers in 5x7 grid on calibration board.

We validated UVDAR 2 performance in the lab (Fig. 3.3). With correct calibration and optimal settings, the system can localize targets up to 20 meters away under realistic illumination conditions. The precision error can be described as gaussian noise with standard deviation of only 0.1 meters in ideal scenarios. The biggest limitation is the FOV of cameras, which is 160 degrees horizontal and 120 degrees vertical, leaving 40 degree dead angle on the left and right side of the UAV. This can be circumvented by flying at similar altitude and constantly rotating all UAVs in the swarm. 40 degrees-per-second was chosen as the optimal rotation speed, as this speed does not negatively affect UAV control at normal speeds while removing the danger of not localizing targets for long durations and colliding.

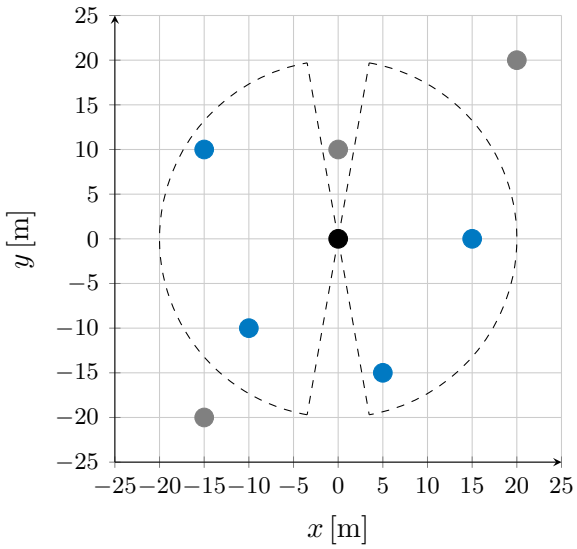


Figure 3.3: Example of UVDAR 2 visibility on RoboFly platform. Top-down view. All UAVs flying at same altitude. Blue dots represent visible targets, grey dots not visible targets.

3.3 Measuring Orientations

Unlike the original UVDAR, the UVDAR 2 does not provide orientation measurements. There are multiple methods being developed in MRS including ML-based vision algorithms, which can estimate the orientation directly from camera feed (Fig. 3.4). While these methods

are promising, they are not yet ready for deployment and developing such methods further is out-of-scope for this thesis. For this reason, measuring orientation is emulated with ROS node sending data over Wi-Fi in the background. The swarming algorithm itself is not aware of this and still acts in fully decentralized fashion as if orientation data were measured locally.

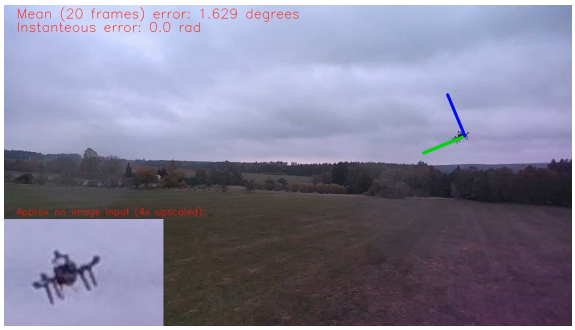


Figure 3.4: Example of vision based algorithm for measuring orientation developed in MRS by Tobias Vinklarek.

3.4 Estimating acceleration

While the proposed swarming model does not require acceleration data, it can be used as input for kalman filters. Although the kalman filter could use orientation itself, this approach allows us to create linear filter, which is easier to implement, less computationally intensive and numerically stable. To estimate acceleration, we need the already measured orientation data in combination with total thrust of the UAV.

First step is computing rotation matrix $\mathbf{R}_{\mathcal{B}}^{\mathcal{W}}$, representing transformation from body frame to world frame. This step depends on the format in which the orientation is represented. Unit quaternion $\mathbf{q} = \left[q_w, q_x, q_y, q_z\right]^\intercal$ is recommended, since it is hard to misinterpret

$$\mathbf{R}_{\mathcal{B}}^{\mathcal{W}} = \begin{bmatrix} 1-2(q_y^2+q_z^2) & 2(q_xq_y - q_wq_z) & 2(q_xq_z + q_wq_y) \\ 2(q_xq_y + q_wq_z) & 1-2(q_x^2+q_z^2) & 2(q_yq_z - q_wq_x) \\ 2(q_xq_z - q_wq_y) & 2(q_wq_x + q_yq_z) & 1-2(q_x^2+q_y^2) \end{bmatrix}.$$

In case of rotation angles roll ϕ, pitch θ and yaw ψ, it is important to specify the order of rotations. In robotics, order $\phi \rightarrow \theta \rightarrow \psi$ is commonly used, the matrix is computed as three consecutive rotations

$$\mathbf{R}_{\mathcal{B}}^{\mathcal{W}} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi),$$

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix},$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix},$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The acceleration is then computed as

$$\mathbf{a} = \frac{1}{m}\mathbf{R}_{\mathcal{B}}^{\mathcal{W}} \begin{bmatrix} 0 \\ 0 \\ f_{th} \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}$$

where m is mass in kg, f_{th} is total thrust in N and g is gravitational constant.

This requires measuring total thrust and while there are some methods being developed at MRS, such as vision-based methods using event cameras, they are not yet proven to be precise enough. If assumption of level-flight at constant altitude is made, the total thrust can be estimated as:

$$mg - \hat{\mathbf{e}}_3^{\intercal}\mathbf{R}_{\mathcal{B}}^{\mathcal{W}} \begin{bmatrix} 0 \\ 0 \\ f_{th} \end{bmatrix} = 0,$$

$$f_{th} = \frac{mg}{\hat{\mathbf{e}}_3^{\intercal}\mathbf{R}_{\mathcal{B}}^{\mathcal{W}}\hat{\mathbf{e}}_3}.$$

The acceleration estimate is not completely accurate as it ignores many physical attributes such as air drag (Fig. 3.5). The estimation could also yield very wrong results, if the UAV is flying against strong winds and has to tilt to maintain position. This however is not an issue for our use-case, since we are only interested in relative acceleration between observer and target, so if we assume that both experience similar wind currents (fair assumption as they are close), this effect cancels out. The estimate is accurate-enough for many use-cases, such as input for kalman filters.

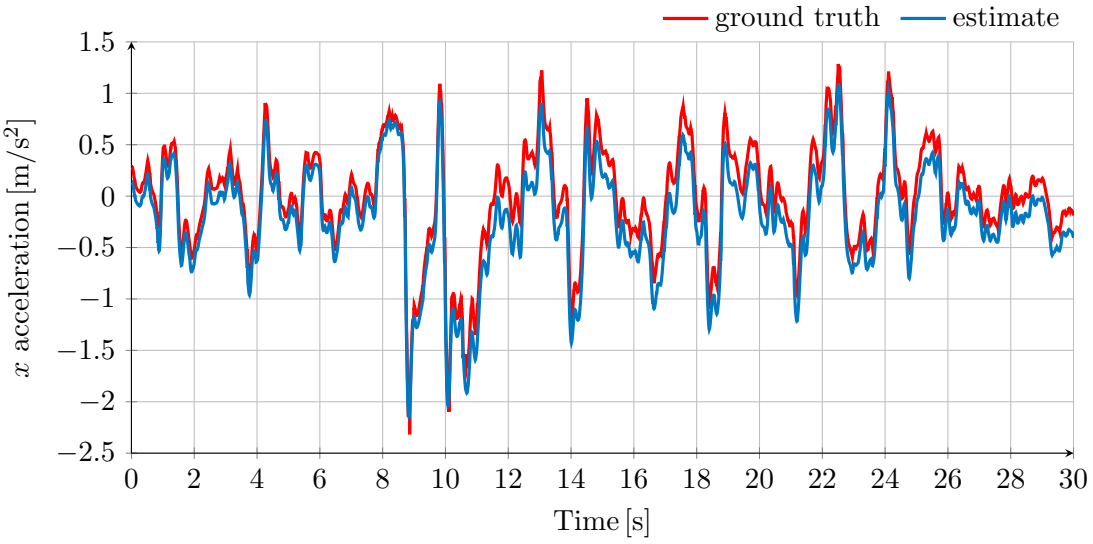


Figure 3.5: Acceleration estimation accuracy. UAV flying at constant altitude, total thrust estimated.

3.5 Position Kalman Filter

A Kalman filter is a powerful algorithm that fuses noisy sensor measurements with a physical model predicting how state evolves over time. At each step it predicts the new state based on the previous state, then adjusts that new state using the newest measurement,

weighting each according to its uncertainty. By continuously repeating this predict-update cycle, the filter effectively smooths out noise and produces more accurate estimate.

Model Definition

First we define the internal state $\mathbf{x}$ of the physical model

$$\mathbf{x} = \left[p_x, v_x, a_x, p_y, v_y, a_y, p_z, v_z, a_z\right]^\intercal,$$

where $\left[p_x, p_y, p_z\right]^\intercal$ is position, $\left[v_x, v_y, v_z\right]^\intercal$ is velocity and $\left[a_x, a_y, a_z\right]^\intercal$ is acceleration. The used state transition matrix $\mathbf{A}$ represents double integrator for each axis and is defined as

$$\mathbf{A}_{block} = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}(\Delta t)^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{block} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{block} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{block} \end{bmatrix}.$$

We use position and acceleration measurements to correct predictions, measurement vector $\mathbf{z}$ and its mapping matrix $\mathbf{H}$ are defined as

$$\mathbf{z} = \left[p_x, p_y, p_z, a_x, a_y, a_z\right]^\intercal,$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Lastly we introduce covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ representing uncertainty of the physical model and measurements respectively. The values are fine-tuned to work with the constraints of introduced localization systems. The uncertainty of physical model accumulates over time and is thus dependent on time delta between iterations.

$$\mathbf{Q} = \begin{bmatrix} 0.05\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.05\Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05\Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.05\Delta t & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.05\Delta t & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.05\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.05\Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05\Delta t \end{bmatrix},$$

$$\mathbf{R} = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0025 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0025 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0025 \end{bmatrix}.$$

The great thing about this setup is that it not only works for the absolute position filtering, if we assume that all data is relative, the filter still works as expected and is thus ideal for improving robustness of our swarming model.

Iteration

At start, the internal state vector $\mathbf{x}$ is initialized to 0 and covariance matrix $\mathbf{P}$ is initialized as identity matrix. The first step, the prediction step, involves predicting the next state and covariance

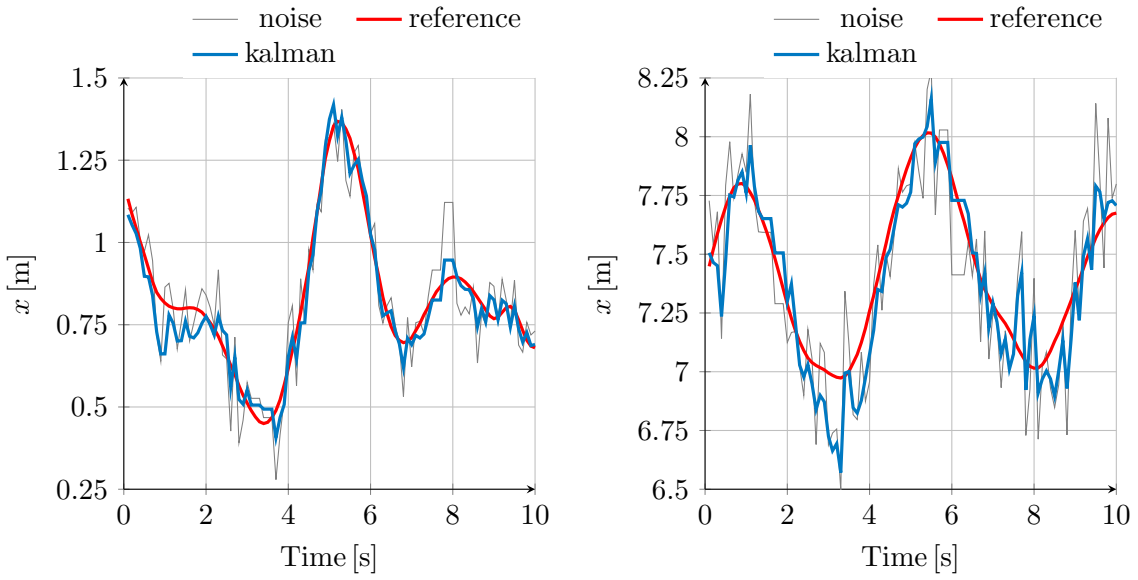
$$\mathbf{x}_k^{'} = \mathbf{A}\mathbf{x}_{k-1},$$
$$\mathbf{P}_k^{'} = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\intercal + \mathbf{Q}.$$

The second step, the update step, computes kalman gain and corrects the state and covariance accordingly

$$\mathbf{K}_k = \mathbf{P}_{k-1}\mathbf{H}^\intercal(\mathbf{H}\mathbf{P}_{k-1}\mathbf{H}^\intercal + \mathbf{R})^{-1},$$
$$\mathbf{x}_k = \mathbf{x}_k^{'} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^{'}),$$
$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^{'}.$$

Accuracy

Selected parameters of the filter give large weight to measurements, meaning that the output is not completely smooth (Fig. 3.6). This is intentional, because giving too large weights to physical model can result in the internal state not reacting quickly to changes. The filter still effectively filters out the largest deviations in the noise. Kalman filter is tested under real-world conditions, meaning that latency is added to orientation measurements to account for delays in Wi-Fi communications (3.3) and limited FOV (Fig. 3.2) is simulated.



(a) std-dev = 0.1 (expected noise)

(b) std-dev = 0.25 (extreme noise)

Figure 3.6: Kalman filter accuracy under real-world conditions.

4 Implementation and Evaluation Metrics

In this chapter we go over implementation details, introduce simulator used for testing and propose metrics for evaluating swarm performance.

4.1 MRS UAV System

The MRS UAV System is an open-source software framework developed by the Multi-Robot Systems Group at Czech Technical University in Prague, designed to assist researchers in advanced R&D in autonomous UAV systems, from speed racing and decentralized swarming to GNSS-denied coordination of multi-UAV formations. [1]

The swarming algorithm and estimation pipeline introduced in previous Chapters 2,3 is implemented in C++ in ROS1 [20] and relies on MRS UAV System [8] (Fig. 4.1). The MRS UAV System is a comprehensive framework providing the core software components used throughout this thesis. This includes custom messages, UAV controllers, state estimators, UAV simulators, reference frame transformation utilities and more.

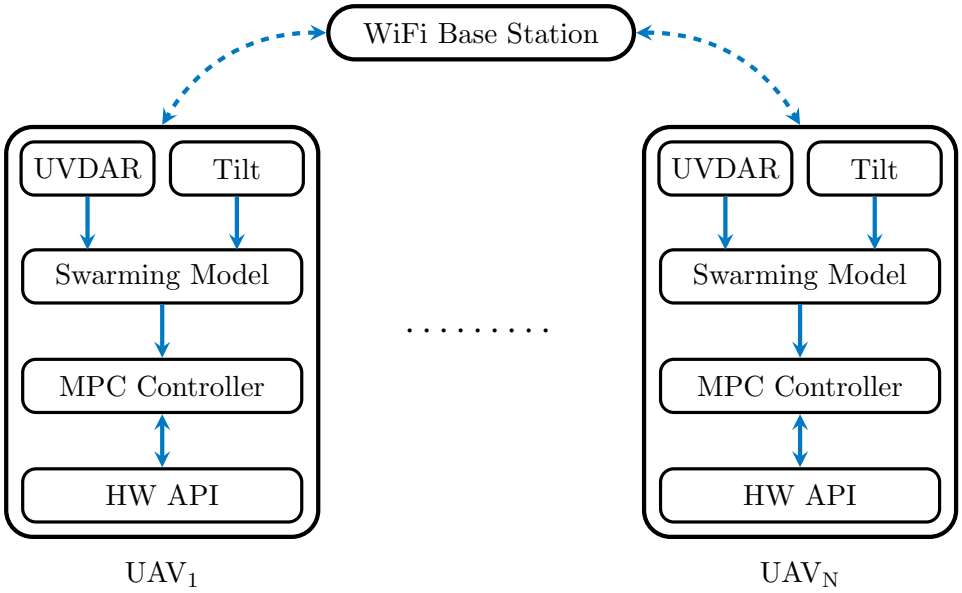


Figure 4.1: Diagram of simplified UAV setup.

4.2 Simulator

Multirotor Simulator (Fig. 4.2), part of the MRS UAV system, is the simulator of choice for this thesis. It strikes the perfect balance between simulation precision and speed. While not as precise as Gazebo Simulator[2], it allows for real-time simulations with hundreds of UAVs. This is crucial for testing and fine-tuning the swarming algorithm. The simulator itself does not provide GUI, so RVIZ[3] is used for visualization.

[1] Official MRS UAV System documentation: https://ctu-mrs.github.io/.
[2] Official Gazebo site: https://gazebosim.org/home.
[3] RVIZ details: https://wiki.ros.org/rviz.

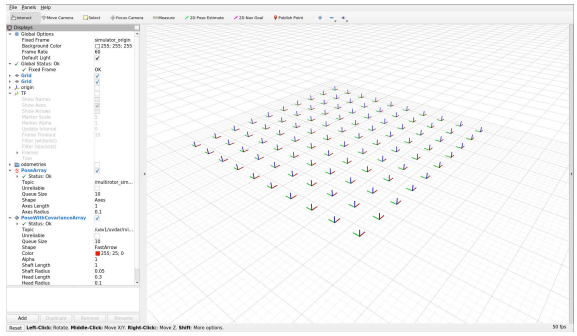


Figure 4.2: MRS Multirotor Simulator, visualized with RVIZ.

4.3 Evaluation Metrics

In this thesis, we mainly focus on swarm stability, which is crucial for real-world deployments, because even small instabilities can trigger chaotic behaviour and cause the formation to break apart. To assess stability, we evaluate separation and alignment of the swarm.

Separation

Separation is the most important factor for us when evaluating swarms, since preventing expensive and dangerous UAV collisions is of high priority. To properly evaluate separation, we propose three main metrics - average distance d_{avg}, minimum distance d_{min} and maximum distance d_{max}.

The average distance describes separation distance between observer and its closest target averaged over all observers in the swarm. The minimum distance describes the minimum distance between any two UAVs in the swarm. In contrast, maximum distance describers maximum distance between any observer and its closest target. They are mathematically defined as

$$d_{avg} = \frac{1}{N} \sum_{b \in B} \min_{\mathbf{t} \in T_b} (\|\mathbf{t}\|),$$

$$d_{min} = \min_{b \in B} \min_{\mathbf{t} \in T_b} (\|\mathbf{t}\|),$$

$$d_{max} = \max_{b \in B} \min_{\mathbf{t} \in T_b} (\|\mathbf{t}\|),$$

where N is a number of observers, B is a set of observers and T is set of target positions relative to given observer. With these metrics, we also define metrics d_{range} and d_{diff}

$$d_{range} = (d_{min}, d_{max}),$$

$$d_{diff} = d_{max} - d_{min}.$$

Together these metrics provide valuable insight into separation stability within the swarm. The average distance d_{avg} gives us idea how the swarm behaves as a whole, minimum distance d_{min} helps us understand if there is instability somewhere in the swarm, and maximum distance d_{max} shows if the whole swarm is cohesive. The minimum distance d_{min} should never approach the minimum safe distance set by parameter d_{safe} (Section 2.2.1). Furthermore, the d_{diff} should not be large.

- **Alignment**

For evaluating alignment, we use metric based on polarization function mathematically defined as

$$p = \left\| \frac{1}{N} \sum_{i=1}^{N} \begin{cases} \hat{\mathbf{v}}_i, & \|\mathbf{v}_i\| \geq 0.1 \\ \hat{\mathbf{e}}_z, & \text{otherwise} \end{cases} \right\|,$$

where N is number of UAVs in the swarm and $\mathbf{v}$ is velocity of UAVs. The resulting number p ranges from 0 to 1, where 0 means that all UAVs within the swarm are flying into opposite directions, 1 in contrast represents all UAVs flying into the same direction.

When the swarm is just hovering around some point, it can sometimes become so stable that it is almost not moving at all and only minor insignificant velocities are observed as a result. These velocities are not aligned and can be considered as just noise. This however can then result in terrible polarization values, even though the swarm is stable and perfectly aligned. To overcome this, every stationary UAV with velocity below threshold 0.1 m/s has its velocity set to vector $\hat{\mathbf{e}}_z$. As a result, all stationary UAVs are considered perfectly aligned.

5 Baseline Comparison

In this chapter, the new swarming algorithm proposed in Chapter 2 is compared against the baseline boids swarming model in simulations with large number of UAVs. Both approaches are tested under ideal conditions, meaning that no noise is introduced to position, orientation and velocity measurements. Because of that, the estimation pipeline proposed in Chapter 3 is not used, position and orientation (velocity in case of baseline) data is used directly without any further filtering. We do not simulate the whole control stack, we only simulate UAV dynamics, which allows us to test large swarms. The testing is organized into two parts:

- **2D swarms** - All UAVs are flying at the same constant altitude.
- **3D swarms** - UAVs do not have fixed altitude.

Within each part, we consider two scenarios:

- **Outbound goal** - All UAVs head freely toward a distant common goal. Because each observer experiences similar tracking force, this scenario is relatively easy to manage.
- **Central goal** - The goal is placed at the swarm's centre, causing UAVs to converge inward. This compression often leads to erratic behaviour, making it one of the most challenging scenarios the swarm can come across. While this scenario basically never occurs in practice (we would decrease target force as goal is approached), it allows us to test the limits of both models.

5.1 Baseline Implementation

Because Reynolds' original Boids paper leaves the precise formulation of each rule open to interpretation, it is essential to specify our baseline in detail. In this thesis, the baseline model retains the same separation, cohesion, and tracking rules as the proposed model, but replaces our novel tilt-matching rule with the traditional velocity-matching rule:

$$\mathbf{f}_a = \frac{\sum_{t=1}^{N} \mathbf{v}_t}{N},$$

where $\mathbf{v}$ is velocity of the target in observer-local-frame (relative velocity). This approach is similar to that in MRS Boids implementation [12, 15].

5.2 Algorithm weights

Weight	Baseline	Enhanced
w_s	5.0	5.0
w_c	1.0	1.0
$\mathbf{w_a}$	**1.5**	**10.0**
w_t	2.5	2.5
r	10.0	10.0
z	1.5	1.5
d_{safe}	5.0	5.0

Table 5.1: Weights used in baseline comparison.

The alignment weights differ because the alignment rule is fundamentally different, other weights are the same (Table 5.1). Fine-tuning these values is difficult, iterative process. Proposed value result in swarm achieving similar speeds. Maximum visibility distance is set to 20 m replicating UVDAR capabilities.

5.3 2D Swarms

In this first evaluation against the baseline Boids algorithm, we deploy a swarm of 20 UAVs constrained to a common, constant altitude.

Outbound Goal

In this easier scenario, both version work almost identically (Fig. 5.1). Both keep safe separation distances and stay perfectly aligned. There are only minor fluctuations in the baseline minimum distances, indicating small instability within the swarm.

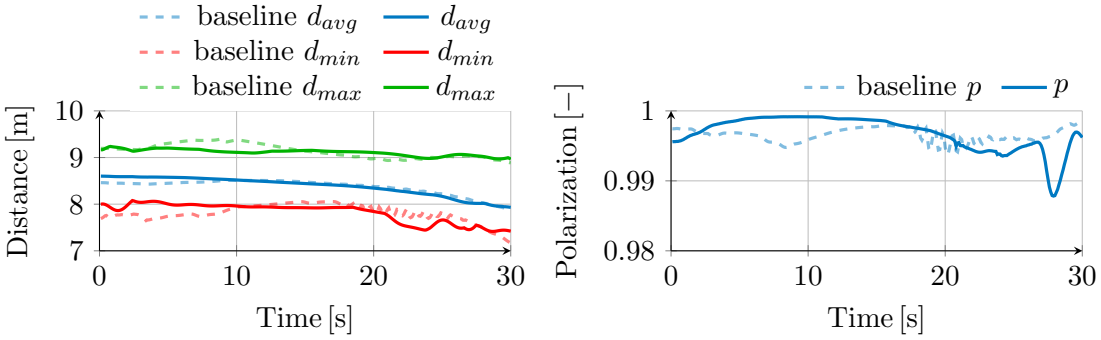


Figure 5.1: Swarm of 20 UAVs flying towards common distant goal at fixed altitude.

Central Goal

In central goal, we can see clear differences (Fig. 5.2, 5.3). As the swarm compresses, the baseline algorithm separation starts oscillating and polarization quickly drops. In contrast, the enhanced model has more stable separation with smaller variance between d_{min} and d_{max}. The enhanced model also manages to hold better polarization for much longer, but as small instabilities develop, the polarization and overall stability drops towards the end.

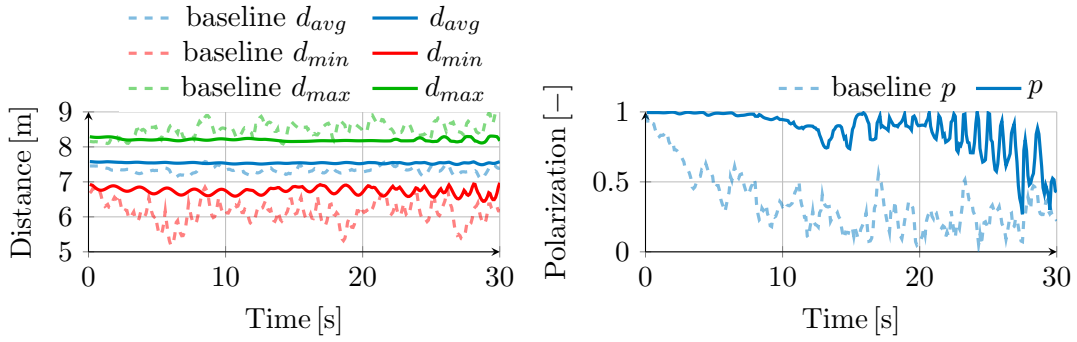
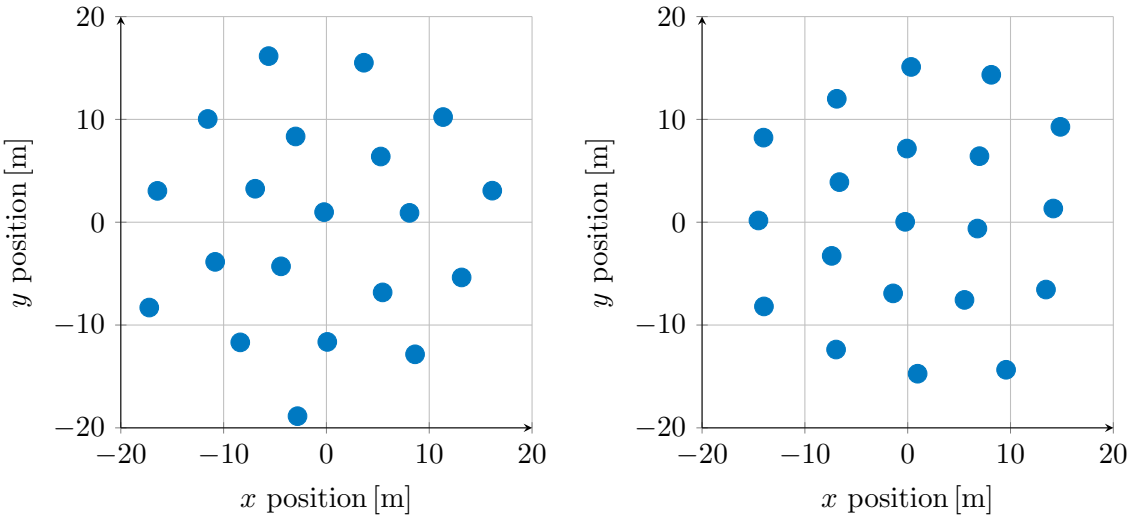


Figure 5.2: Swarm of 20 UAVs flying towards common central goal at fixed altitude.



(a) Baseline (b) Enhanced

Figure 5.3: Formation of 20 UAVs at the end of Fig. 5.2. Goal at coordinates $x = 0$m, $y = 0$m.

5.4 3D Swarms

Extending the comparison to three dimensions, we simulate a larger formation of 100 UAVs with unrestricted altitude, allowing the swarm to form into complex structures.

Outbound Goal

The results (Fig. 5.4) for this scenario mirror 2D results. Both models are stable, have good separation and are perfectly aligned. Measurements started few seconds after initialization of both models and since the baseline model took more time to get into final formation, baseline d_{max} is higher at the start.

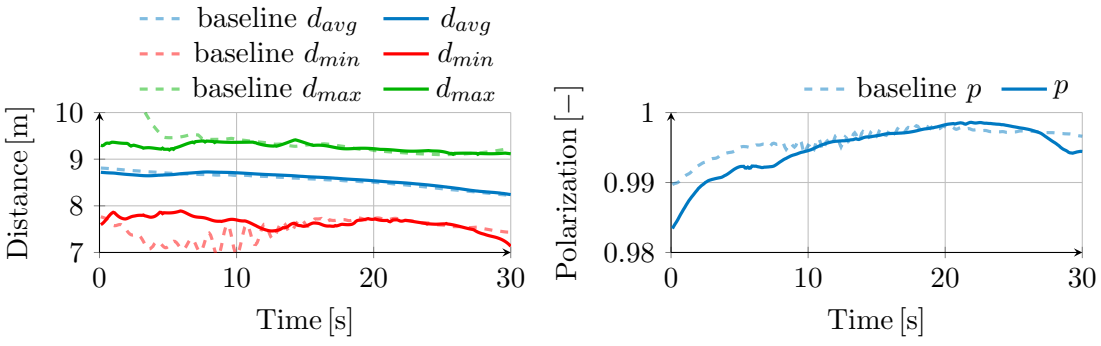


Figure 5.4: Swarm of 100 UAVs flying towards common outbound goal.

Central Goal

Results for this scenario showcase the largest improvement over the baseline (Fig. 5.5, 5.6). The enhanced model has perfect separation and holds almost perfect alignment.

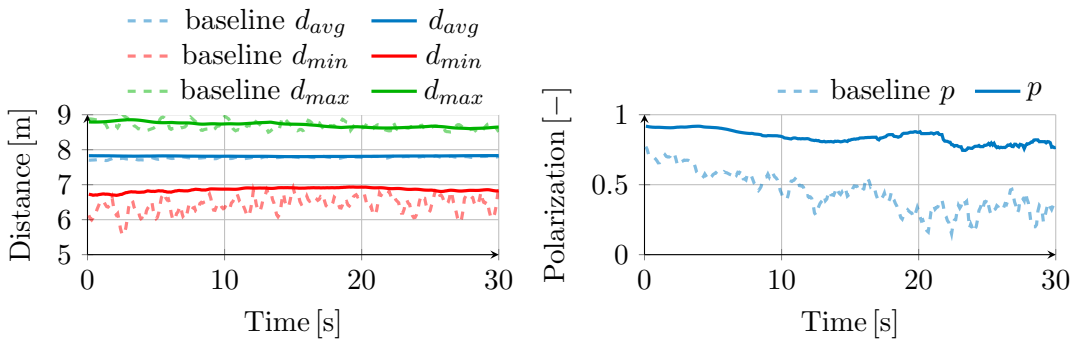
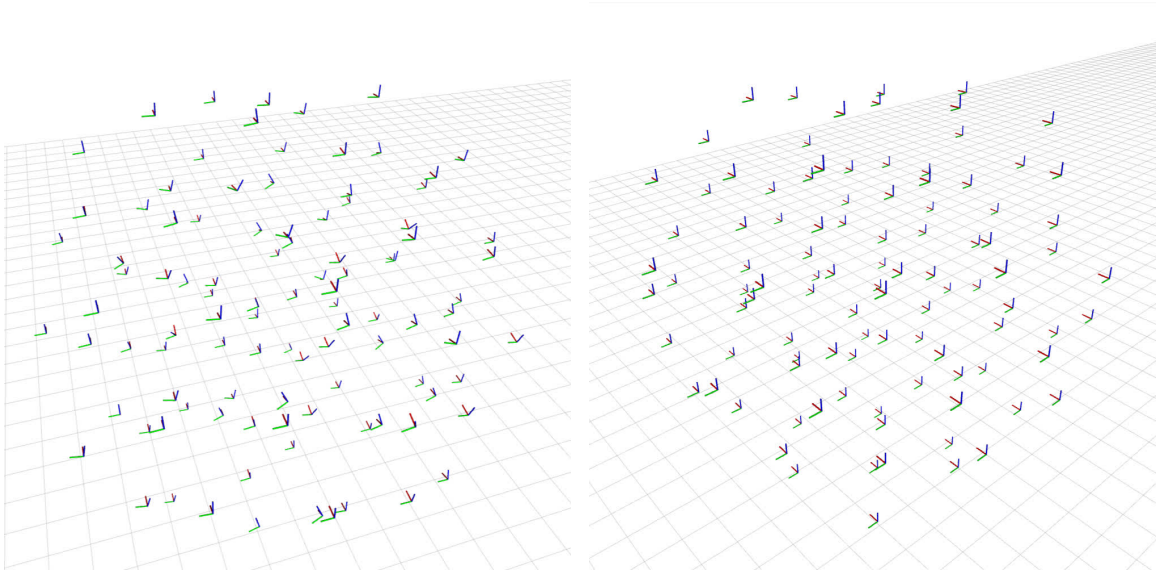


Figure 5.5: Separation evaluation. Swarm of 100 UAVs flying towards common central goal.



(a) Baseline (b) Enhanced

Figure 5.6: Formation of 100 UAVs at the end of Fig. 5.5. RViz used for visualization.

5.5 Summary

The results shown in this chapter showcase the great potential of the proposed model (Table 5.2). While both worked well and neither ever approached dangerous separation distances, the new model matched or outperformed the baseline in every scenario, with major improvements especially in the central goal scenarios. The proposed model also performed well in 3d scenarios, proving that the new tilt-matching rule is well suited not only for 2d swarms, but also complex 3d swarms. Crucially, all these gains are achieved without requiring hard-to-measure target velocity data, only requiring the directly-measurable position and orientation data.

Scenario	Model	$\mu(p)$	$\sigma(p)$	$\sigma(d_{avg})$	$\sigma(d_{diff})$	$\mu(d_{diff})$
2D out.	baseline	1.00[−]	0.00[−]	0.16[m]	0.20[m]	1.33[m]
	enhanced	1.00[−]	0.00[−]	0.20[m]	0.15[m]	1.29[m]
	improvement	**0**[%]	**0**[%]	**−25**[%]	**25**[%]	**3**[%]
2D cent.	baseline	0.35[−]	0.20[−]	0.10[m]	0.39[m]	2.37[m]
	enhanced	0.87[−]	0.15[−]	0.02[m]	0.10[m]	1.46[m]
	improvement	**149**[%]	**25**[%]	**80**[%]	**74**[%]	**38**[%]
3D out.	baseline	1.00[−]	0.00[−]	0.16[m]	0.61[m]	1.98[m]
	enhanced	0.99[−]	0.00[−]	0.14[m]	0.13[m]	1.62[m]
	improvement	**−1**[%]	**0**[%]	**13**[%]	**79**[%]	**18**[%]
3D cent.	baseline	0.44[−]	0.14[−]	0.03[m]	0.29[m]	2.25[m]
	enhanced	0.84[−]	0.05[−]	0.01[m]	0.11[m]	1.86[m]
	improvement	**91**[%]	**64**[%]	**67**[%]	**62**[%]	**17**[%]

Table 5.2: Statistics for the simulated swarm. Population standard deviation used.

6 Full System Simulation

In this chapter the new swarming model introduced in Chapter 2 is tested in simulator under real-world conditions. We simulate all limitations of localization systems described in Chapter 3. All APIs running onboard of real UAV are also simulated, such as hardware API and flight controllers.

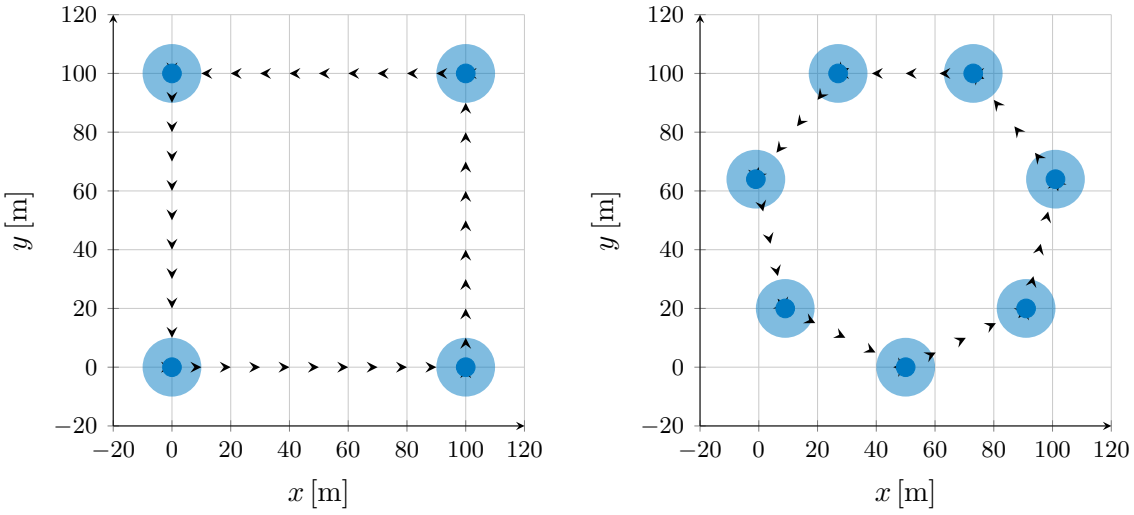
We focus on smaller swarms consisting of 3 and 5 UAVs flying at same constant altitude, which allows us to estimate acceleration just with orientation data as explained in Section 3.4. All UAVs within the swarm are running kalman filters as described in Section 3.5 to improve the accuracy of noisy position measurements.

6.1 Testing Setup

To test the swarm, we consider a realistic scenario where the swarm follows pre-defined path, which is defined as sequence of points. The swarm flies point-to-point until goal is reached. Each point can be considered as point-of-interest, which can be localized by vision-based methods. For simplicity, points along the path are located via GNSS in this thesis.

Because of the decentralized nature of the model, every UAV has its own copy of the path, which it follows on its own. Every UAV can be flying towards different point at any given time. As UAV comes within certain radius of its current point along the path, it updates to the next point in the sequence.

The difficulty of the path is mainly determined by how much UAVs have to turn when switching to next point on the path. We consider two shapes (Fig. 6.1) - square path and heptagon path. The square path represents the more difficult case, where UAVs have to make 90-degree turns.



(a) Square path.

(b) Heptagon path.

Figure 6.1: Visualization of paths. Circles around points represent area (10 m radius), in which UAVs switch to next point along the path.

6.2 Algorithm Weights

For this chapter, we choose slightly different weights optimized for smaller swarms flying in 2d formations. Proposed parameters (Table 6.1) result in swarm speed around 3-4 m/s when flying between points along the path.

Weight	Value
w_s	2.5
w_c	1.75
w_a	7.5
w_t	3.0
r	10.0
z	1.25
d_{safe}	3.0

Table 6.1: Weights used in simulation and deployment.

6.3 Controller Constraints

Since we are simulating the whole control stack, we have to specify MPC controller constraints. While our model itself can provide any velocity reference to the controller, the controller can limit maximum speed, acceleration, rotation and so on. Following constraints (Table 6.2) were used, which were optimized to work well with model weights (Table 6.1):

Constraint	Value
horizontal speed	4.0[m/s]
horizontal acceleration	1.5[m/s^2]
horizontal jerk	40.0[m/s^3]
yaw	40.0[deg/s]
tilt	60.0[deg]

Table 6.2: Controller constraints used in simulation.

6.4 3 UAVs

We start by evaluating the smallest possible swarm - group of 3 UAVs. Since the swarm is small, we set radius around points along the path to 9 m, with the final point being exception at 12 m. This is to ensure better swarm stability when coming to a stop at the end of path.

Square Path

When flying along the square path, the small swarm manages to maintain safe separation while being cohesive (Fig. 6.2, 6.3, Table 6.3). Results clearly show when the swarm arrives to points along the path, specifically at times 35, 65, 95, 125 s. This is in almost perfect intervals of 30 seconds, which is the result of relatively constant speed. It also represents the only times, when the swarm loses perfect polarization and distances between UAVs lower. During the short periods at the start and at the end (0-10, 125-140 s), the swarm just hovers in place resulting in very low speeds and not ideal polarization values.

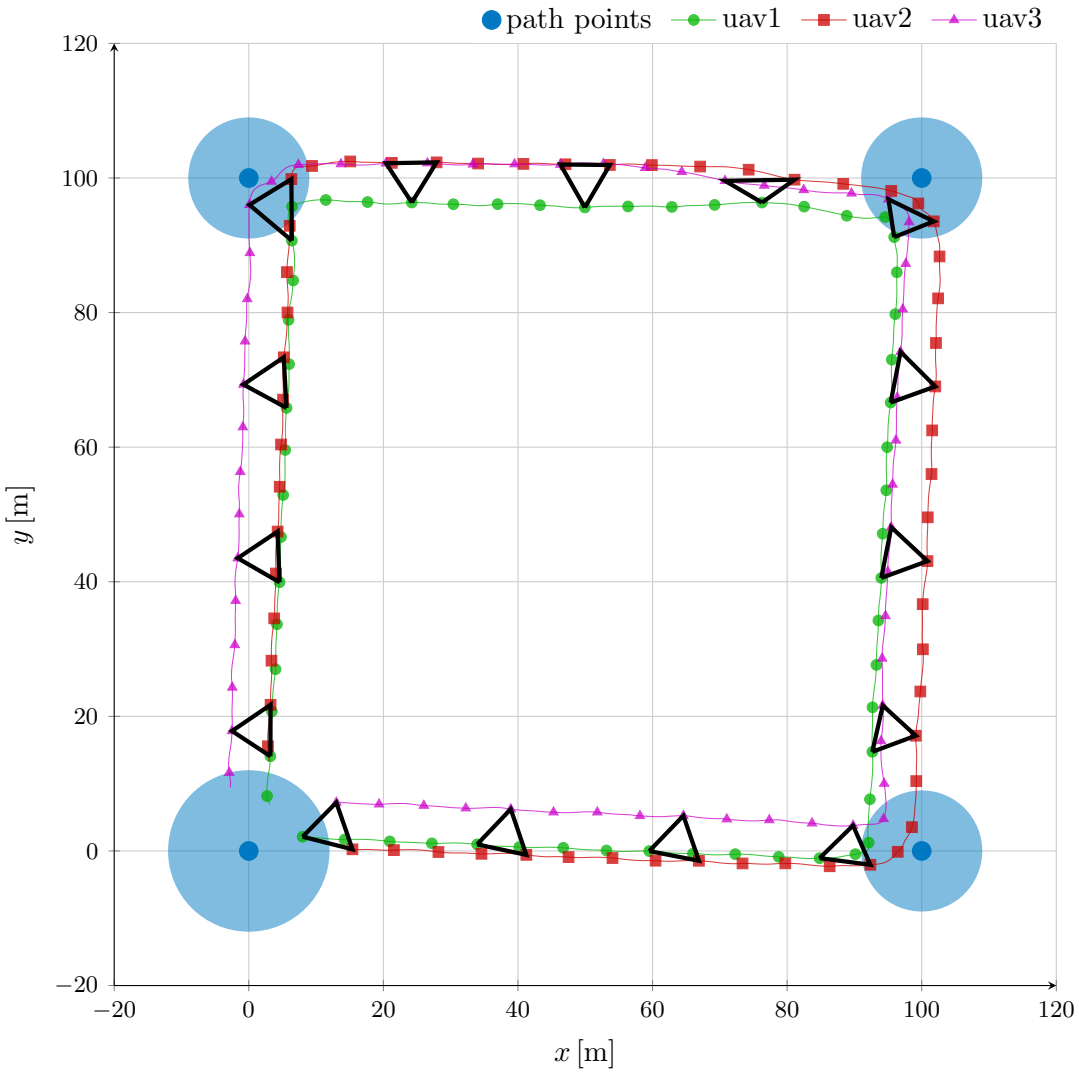


Figure 6.2: Swarm of 3 UAVs. Flying along square path (6.1a). Marker every 2 s. First and last few seconds of flight removed for readability.

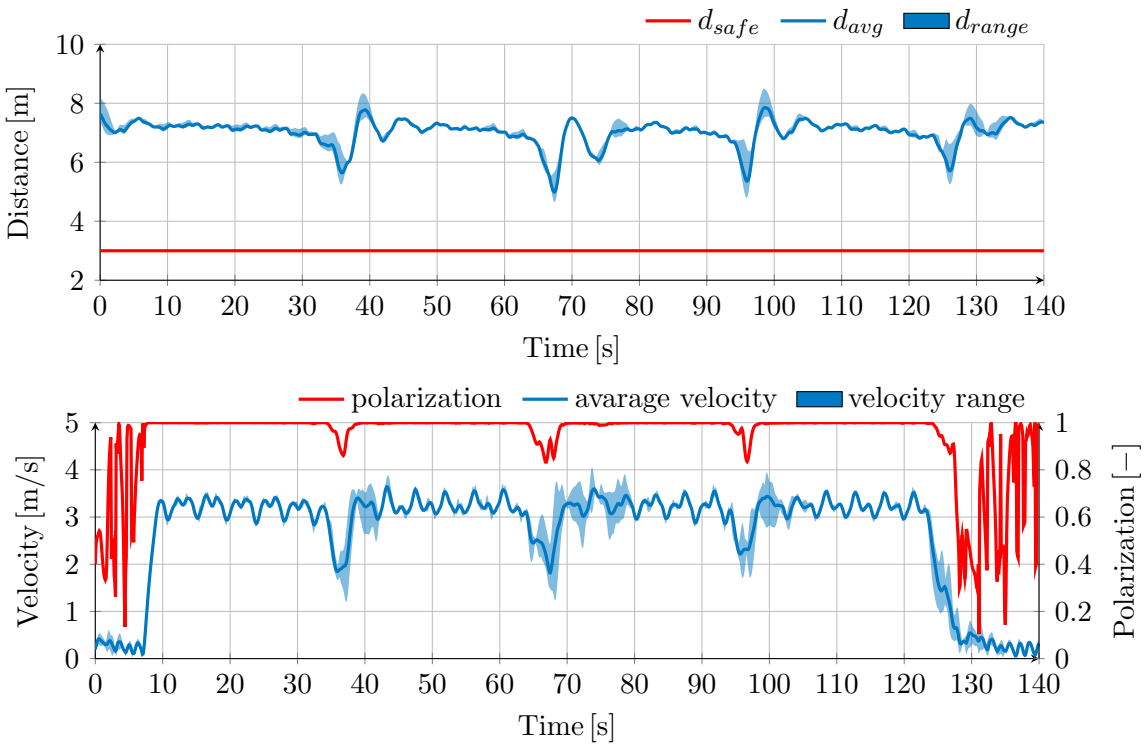


Figure 6.3: Swarm of 3 UAVs. Flying along square path (6.1a).

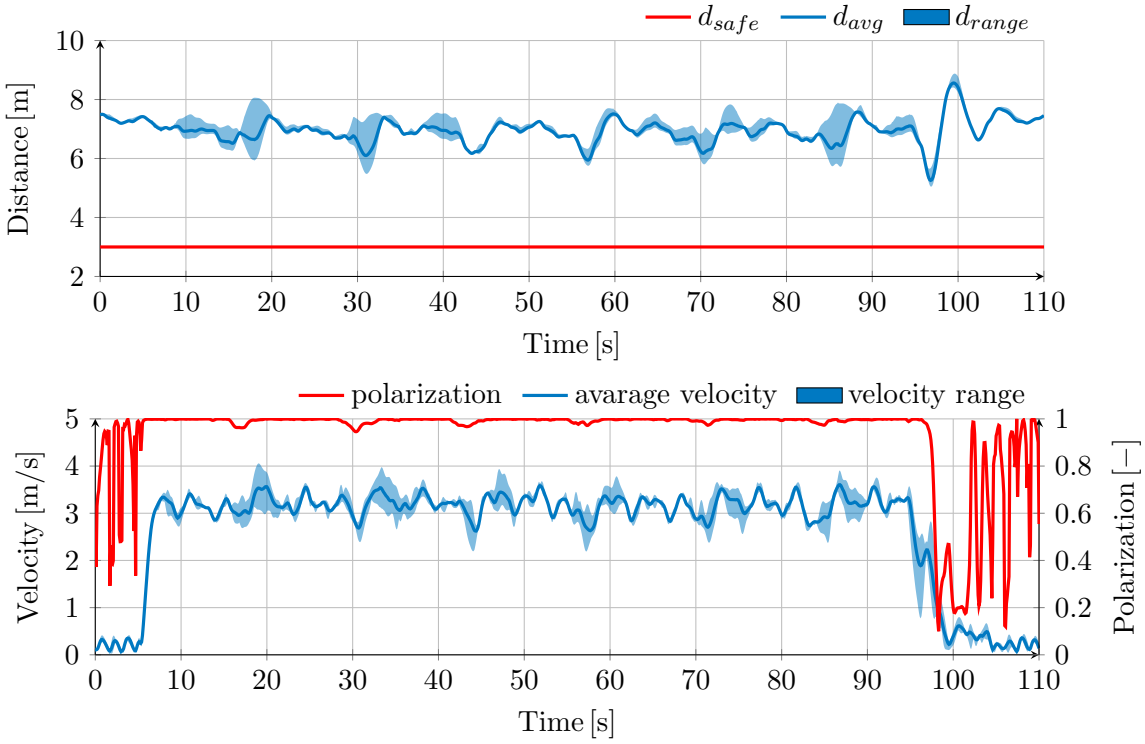


Figure 6.4: Swarm of 3 UAVs. Flying along heptagon path (6.1b).

Heptagon Path

The small swarm performs even better on heptagon path (Fig. 6.4, 6.5, Table 6.3), showcasing that the smaller changes in direction make this path easier than the square path. It is still easy to see when swarm arrives to points along the path at times 16, 29, 42, 55, 70, 83, 96 s, but both polarization and distances between UAVs are affected much less. Velocity reaches similar levels of square path test, although with much less pronounced variations.

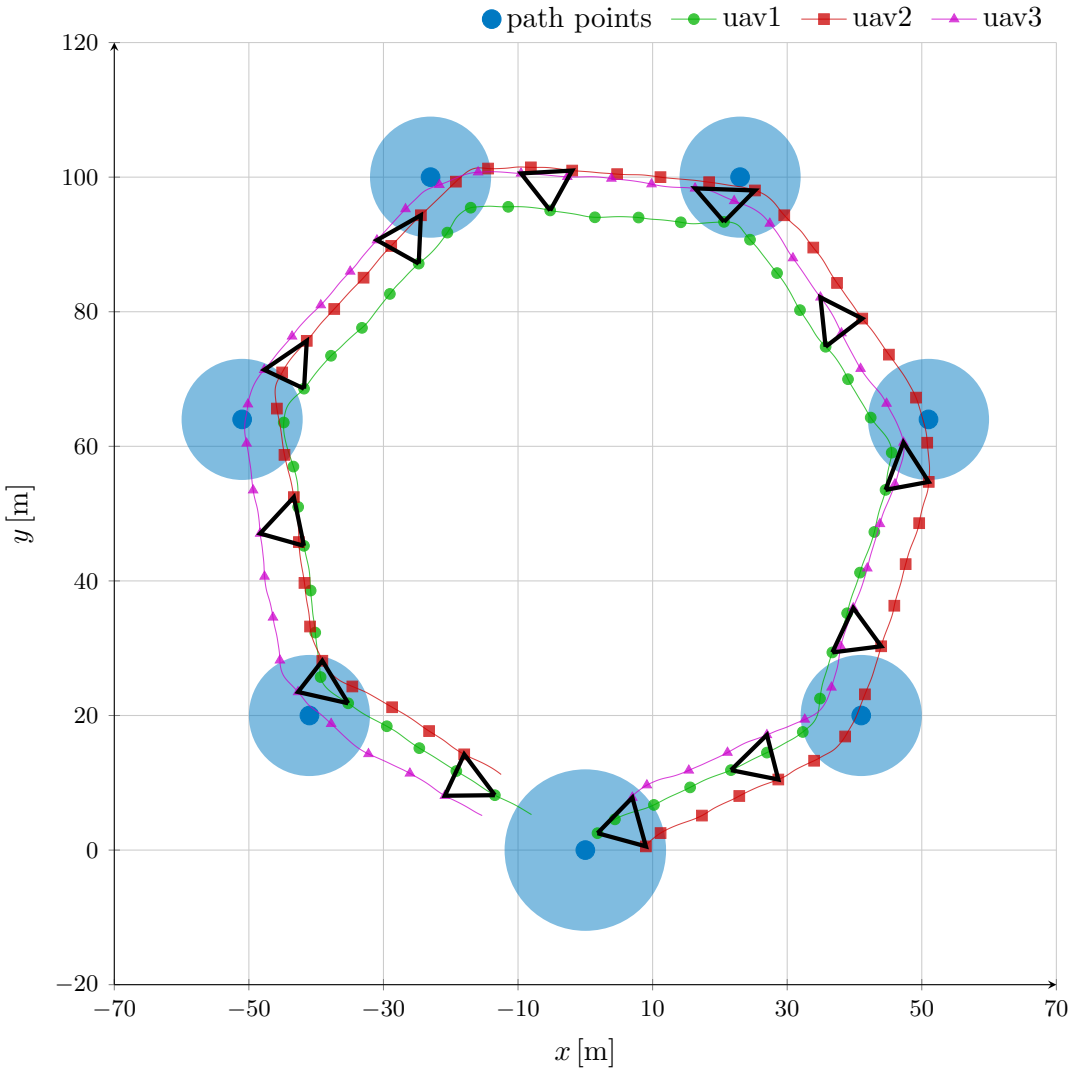


Figure 6.5: Swarm of 3 UAVs. Flying along heptagon path (6.1b). Marker every 2 s. First and last few seconds of flight removed for readability.

6.5 5 UAVs

We now deploy larger swarm consisting of 5 UAVs. This allows us to better understand the scalability of the model, while still being easy to visualize. Because the swarm is now larger, we set radius around points along the path to 15 m, with the final point being exception at 20 m.

Square Path

Same as with the smaller swarm, the larger swarm manages to maintain safe distances while being cohesive (Fig. 6.6, 6.7, Table 6.3). Drops in separation and polarization can again be observed in intervals of around 30 s as UAVs approach points along the path. Drops in polarization are more pronounced than in smaller swarms, which is expected, because there is large time difference between first and last UAV approaching the point, causing larger misalignment. The minimum distance remains same, indicating that the model scales well.

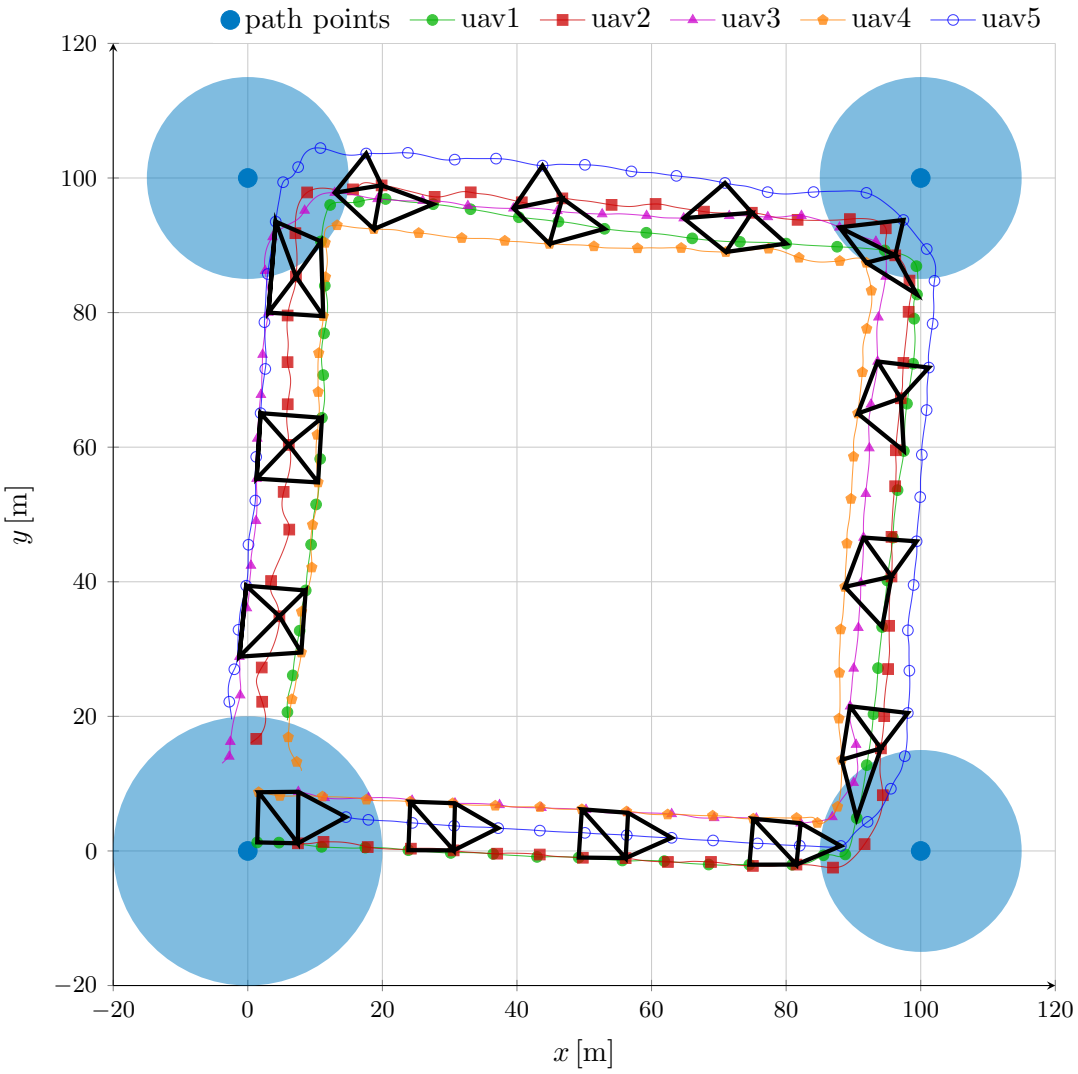


Figure 6.6: Swarm of 5 UAVs. Flying along square path (Fig. 6.1a). Marker every 2 s. First and last few seconds of flight removed for readability.

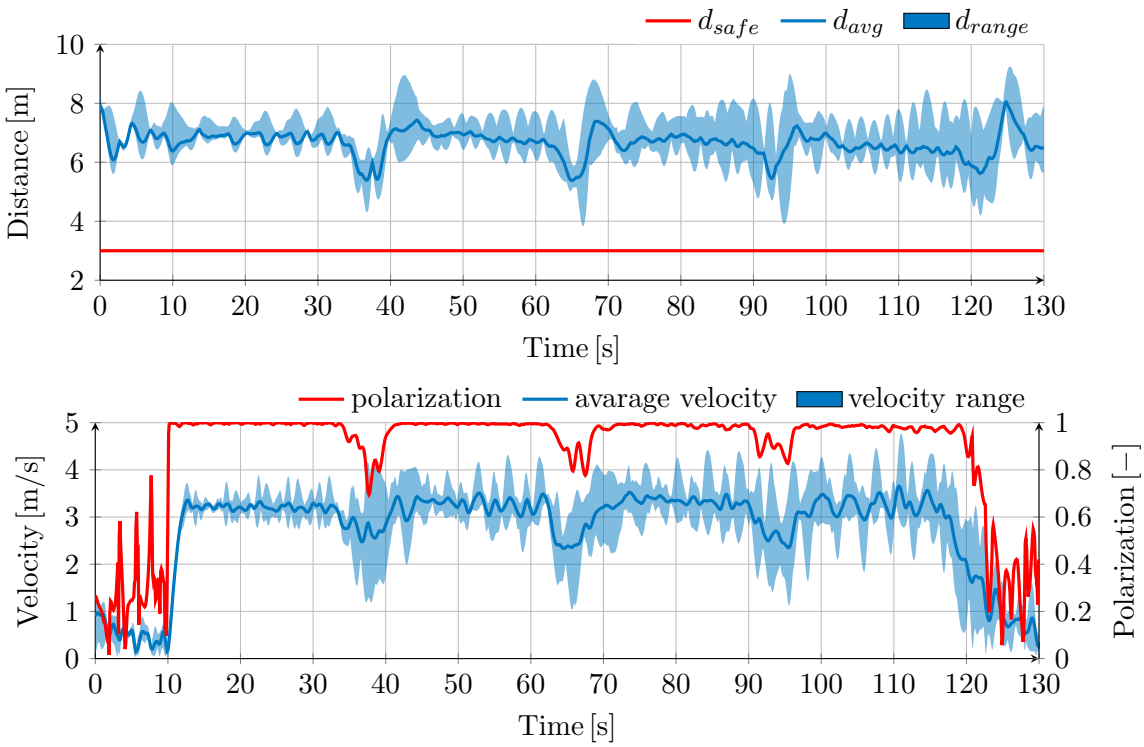


Figure 6.7: Swarm of 5 UAVs. Flying along square path (Fig. 6.1a).

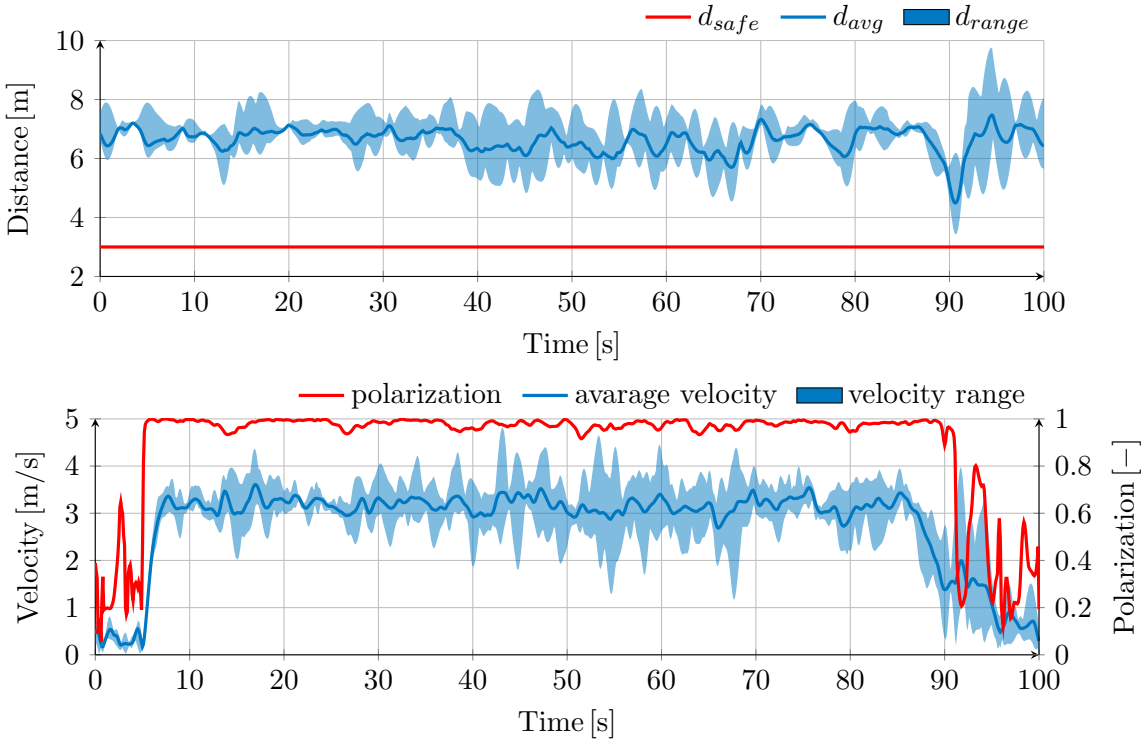


Figure 6.8: Swarm of 5 UAVs. Flying along heptagon path (Fig. 6.1b).

Heptagon Path

As with the smaller swarm, the heptagon path proves to be easier to traverse (Fig. 6.8, 6.9, Table 6.3). Only minor drops in polarization can be seen as swarm arrives to points. There is one significant drop in separation towards the end, as the swarm comes to sudden stop, but it remains above minimum safe level.

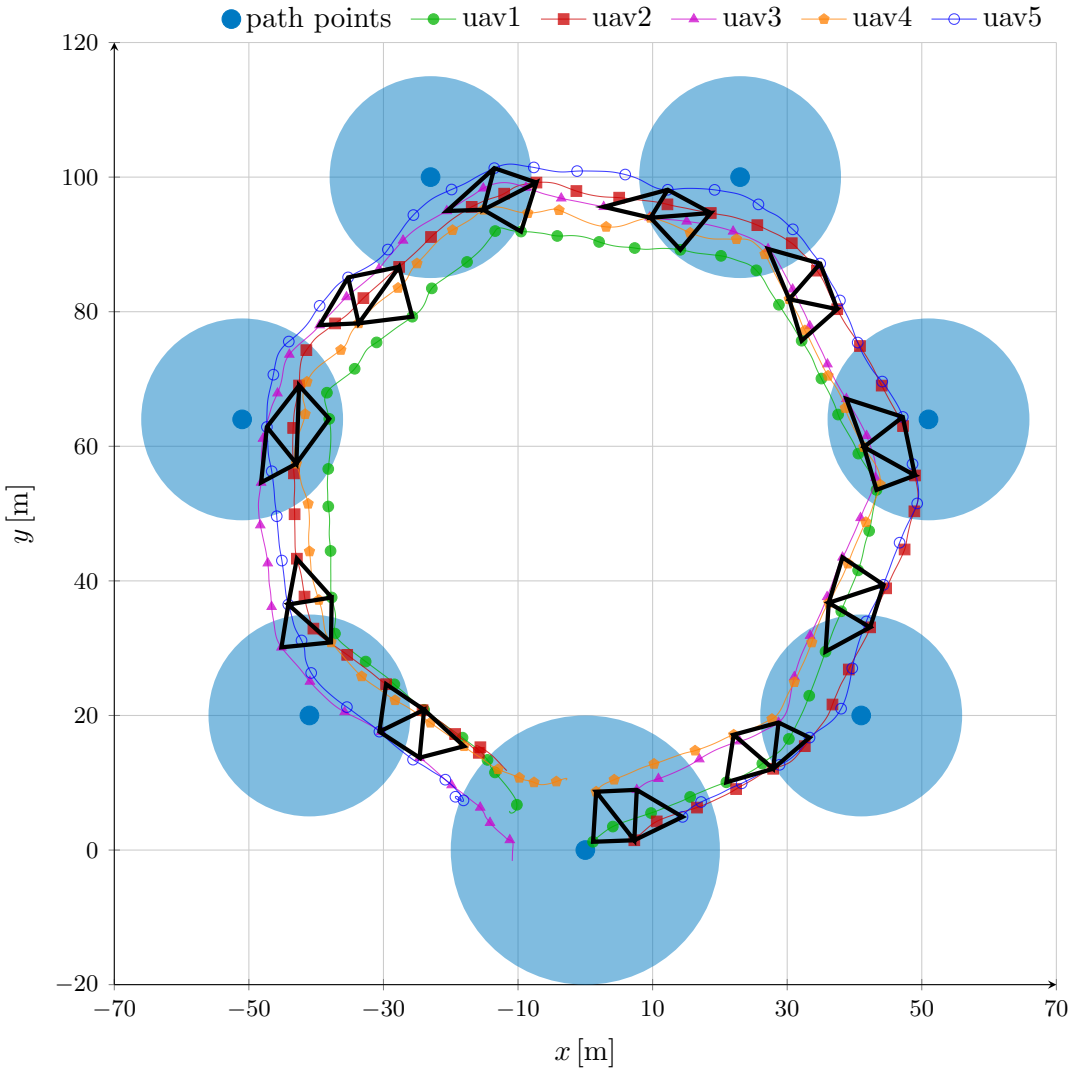


Figure 6.9: Swarm of 5 UAVs. Flying along heptagon path (Fig. 6.1b). Marker every 2 s. First and last few seconds of flight removed for readability.

Swarm Size	Path	$\mu(p)$	$\sigma(p)$	$\sigma(d_{avg})$	$\sigma(d_{diff})$	$\mu(d_{diff})$
3	square	0.94[−]	0.15[−]	0.43[m]	0.29[m]	0.25[m]
	heptagon	0.94[−]	0.18[−]	0.42[m]	0.44[m]	0.41[m]
5	square	0.88[−]	0.24[−]	0.43[m]	0.84[m]	1.44[m]
	heptagon	0.89[−]	0.22[−]	0.40[m]	0.84[m]	1.38[m]

Table 6.3: Statistics for the simulated swarm. Population standard deviation used.

7 Deployment

In this chapter, we validate the results from previous Chapter 6 on real-world experiments. We focus particularly on smaller swarms consisting of 3 UAVs flying along the heptagon path, as shown in simulation Section 6.4.2. We use the same algorithm weights as in previous Chapter 6.2.

7.1 Hardware Platform

Due to hardware issues with Roboflyes, we were forced to switch to different UAV platform at the last minute. The platform used for deployment is X500 [5], a modular research platform developed at MRS (Fig. 7.1). This platform is well established and has been used at MRS for many years. Unfortunately this limits us to swarm of only 3 UAVs.

Another drawback of this platform is that it does not support UVDAR 2 as of time of the deployment, which is crucial part of our estimation pipeline. For this reason, ROS node simulates UVDAR 2 by sending data over Wi-Fi in the background, similar to orientation measurements. As with orientation, the swarming algorithm itself is not aware of the communication in the background and acts in decentralized fashion. GPS by itself would not be accurate enough to simulate UVDAR 2, for this reason RTK base station was also deployed.

Figure 7.1: X500 UAV at Temešvár MRS camp. Not the exact setup used in deployment.

7.2 Controller Constraints

We use more conservative constraints (Table 7.1), so that the risk of failure is low. Unfortunately due to time constraints and bad weather conditions, deployments at full speed shown in Chapter 6 were not possible. These changes mostly affect top speed when flying between points along the path.

Setting	Value
horizontal speed	1.25[m/s]
horizontal acceleration	1.0[m/s^2]
horizontal jerk	40.0[m/s^3]
yaw	40.0[deg/s]
tilt	60.0[deg]

Table 7.1: Controller constraints used in deployment.

7.3 Deployment Results

The results show that, even in the real-world, the swarming model is able to maintain safe separation while being cohesive (Fig. 7.2, 7.3, Table 7.2). The minimum distance between UAVs never approaches minimum safe distance, with margins being even higher than in simulation, which is caused by lower overall speed. While the polarization is not as perfect as in simulation, it is still good.

The results are especially significant, because they were achieved under suboptimal conditions, showcasing the robustness of the proposed model. Severe wind disturbances were observed during flight, which negatively impacted control characteristics of UAVs. Furthermore, the latency spikes in communication reaching up to 2 s were common. We strongly believe that the overall performance would be even better, if deployed under more favourable conditions.

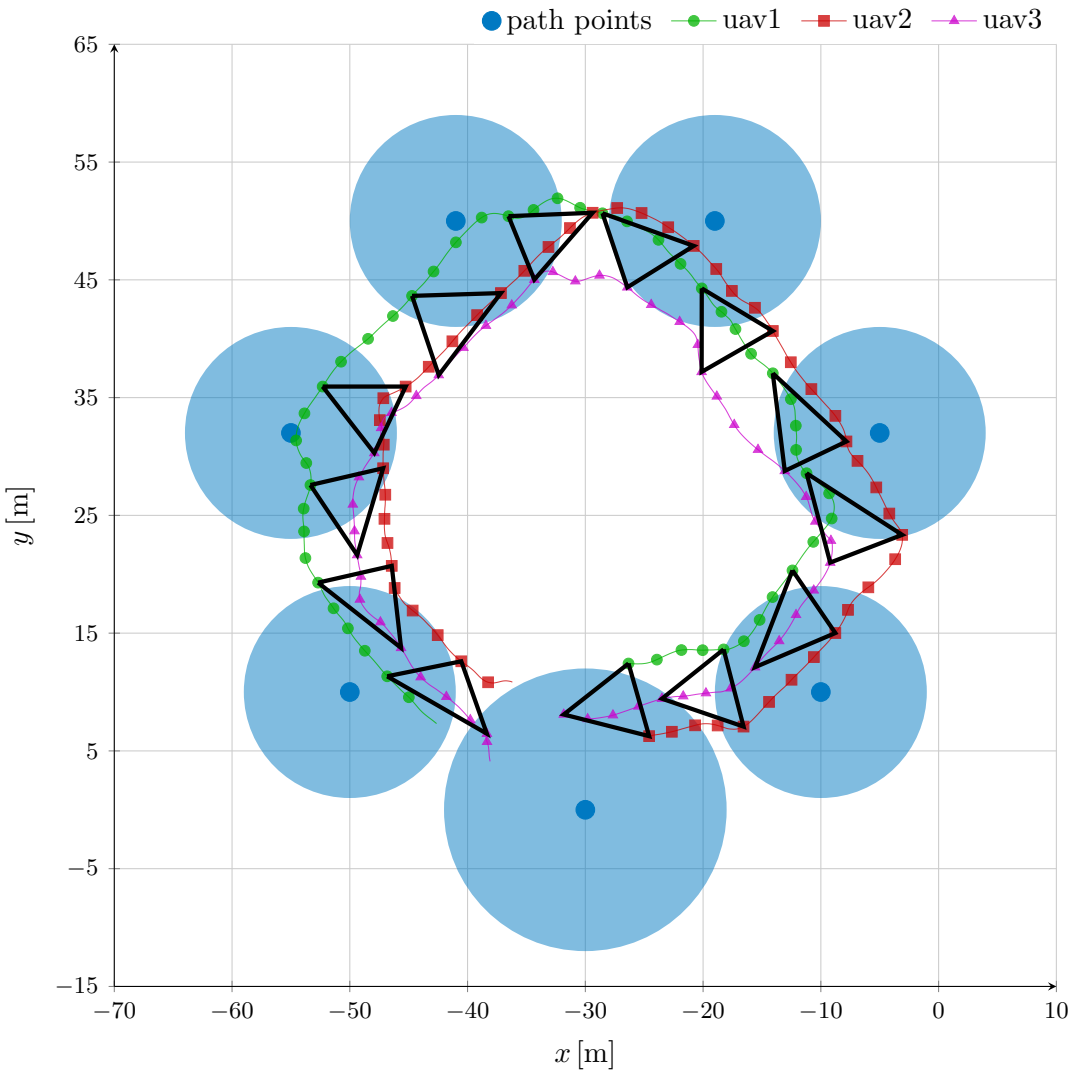


Figure 7.2: Swarm of 3 UAVs deployed in real-world. Flying along heptagon path (6.1b). Marker every 2 s. First and last few seconds of flight removed for readability.

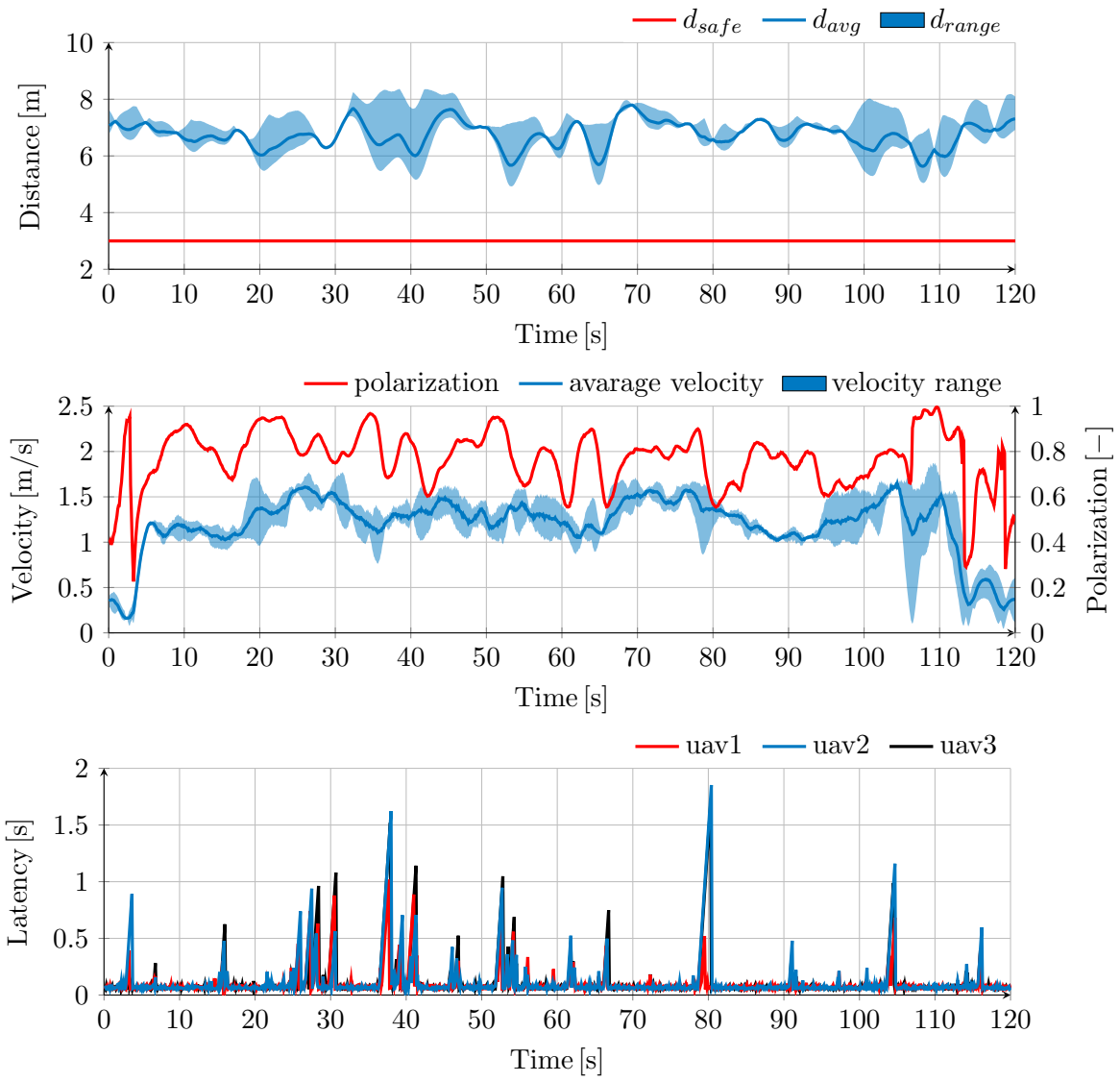


Figure 7.3: Swarm of 3 UAVs deployed in real-world. Flying along heptagon path (Fig. 6.1b).

$\mu(p)$	$\sigma(p)$	$\sigma(d_{avg})$	$\sigma(d_{diff})$	$\mu(d_{diff})$
0.77[−]	0.13[−]	0.44[m]	0.74[m]	0.90[m]

Table 7.2: Statistics for swarm of 3 UAVs. Population variance used.

7.4 Deployment Footage

Footage captured from the real-world deployment can be seen below (Fig. 7.4), showcasing safe separation while being cohesive. Examining videos in attachments (refer to Chapter A) is highly recommended to get better idea of the overall real-world performance. Results shown in this chapter correspond to round 2 footage. Shots from multiple angles are included.

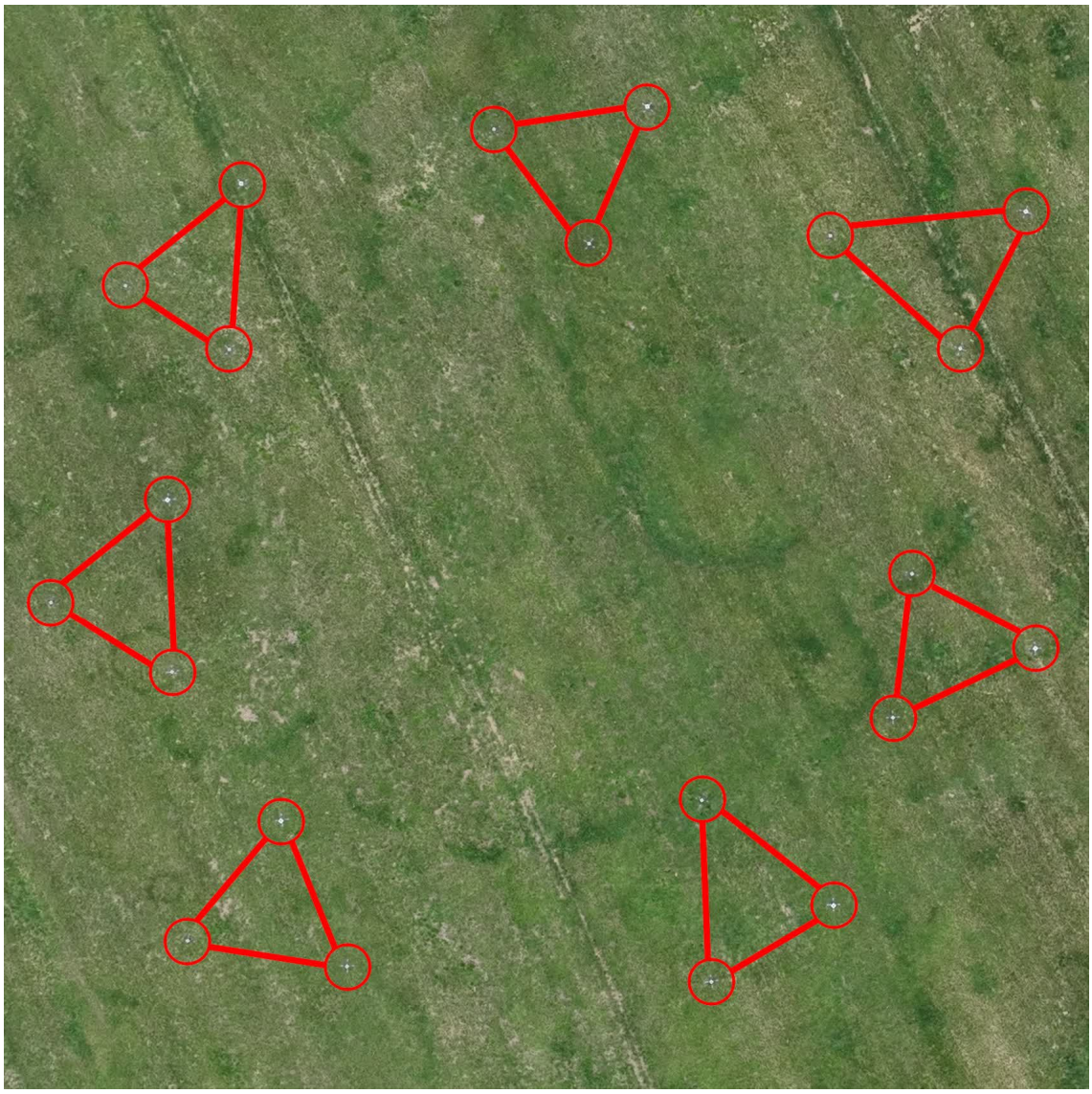


Figure 7.4: Illustration photo merged from multiple video frames. Positions not completely accurate due to camera movement, for precise positions refer to Fig. 7.2. Compared to Fig. 7.2, the image is rotated approximately 45 degrees counter-clockwise. Shot on DJI Mavic 3.

8 Conclusion

In this thesis, we have successfully developed a new fully-decentralized swarming algorithm expanding on the ideas of the original Boids model. The new algorithm was integrated into ROS and MRS UAV System. Moreover, robust estimation pipeline was proposed with heavy focus on real-world deployment.

Simulations on large-scale swarms proved that use of second-and-higher-order data can be beneficial and produce better results then baseline Boids model, especially for real-world deployments. The model was also tested in simulator under real-world conditions, showing the robustness against severe sensor noise. These results were confirmed by real-world deployment on small swarms.

To summerize, following tasks from the assignment were successfully accomplished in this thesis:

- Decentralized swarming was thoroughly studied, with main focus on Boids model.
- The proposed, fully-decentralized model, was fully integrated in ROS and MRS UAV System and written in C++.
- The proposed model replaced traditional velocity-matching function with tilt-matching function, utilizing second-derivative data.
- The proposed model was evaluated against baseline model on large-scale swarms.
- Robust estimation pipeline was developed, and the proposed model was tested under real-world conditions.
- The proposed model was successfully deployed in real-world.

8.1 Future work

This work has shown the potential of the proposed model. We suggest further research in following areas:

- **Agile swarm deployment** - deploy the algorithm without speed limitations. This thesis already proved the viability of higher-speed flight, but only in simulations.
- **Large swarm deployment** - deploy the algorithm with at least 5 UAVs to better evaluate real-world swarm performance.
- **Decentralized localization deployment** - in this thesis, decentralized localization systems were only simulated due to hardware limitations.
- **Dynamic model parameters** - it is impossible to optimize the model parameters, so that they work the best in every possible situation. We believe that system for dynamically optimizing model parameters based on swarm behaviour could greatly improve overall performance.
- **Obstacle avoidance** - development of obstacle avoidance for the proposed model could increase its usability in real-world deployments.

9 References

[1] A. Ahmad, D. Bonilla Licea, G. Silano, T. Baca, and M. Saska, "PACNav: Enhancing Collective Navigation for UAV Swarms in Communication-Challenged Environments," in *2024 IEEE International Conference on Robotics and Automation (ICRA), Contribution accepted for discussion at the workshop session: "Breaking Swarm Stereotypes", Yokohama, Japan*, Contribution accepted for discussion at the workshop session: "Breaking Swarm Stereotypes", Yokohama, Japan, May 2024, pp. 1–2. DOI: 10.48550/arXiv.2404.13440. [Online]. Available: https://doi.org/10.48550/arXiv.2404.13440.

[2] J. Horyna, V. Kratky, V. Pritzl, T. Baca, E. Ferrante, and M. Saska, "Fast Swarming of UAVs in GNSS-denied Feature-Poor Environments without Explicit Communication," *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5284–5291, Apr. 2024. DOI: 10.1109/LRA.2024.3390596.

[3] A. Chaudhary, T. Nascimento, and M. Saska, "Controlling a Swarm of Unmanned Aerial Vehicles Using Full-Body k-Nearest Neighbor Based Action Classifier," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, Jun. 2022.

[4] A. Chaudhary, T. Nascimento, and M. Saska, "Controlling a swarm of unmanned aerial vehicles using full-body k-nearest neighbor based action classifier," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 544–551. DOI: 10.1109/ICUAS54217.2022.9836097.

[5] D. Hert *et al.*, "Mrs modular uav hardware platforms for supporting research in real-world outdoor and indoor environments," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2022, pp. 1264–1273. DOI: 10.1109/ICUAS54217.2022.9836083.

[6] J. Horyna *et al.*, "Decentralized swarms of unmanned aerial vehicles for search and rescue operations without explicit communication," *Autonomous Robots*, pp. 1–17, 2022. DOI: https://doi.org/10.1007/s10514-022-10066-5.

[7] T. Tzanetos *et al.*, "Ingenuity mars helicopter: From technology demonstration to extraterrestrial scout," in *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 01–19. DOI: 10.1109/AERO53065.2022.9843428.

[8] 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 May 2021. DOI: 10.1007/s10846-021-01383-5. [Online]. Available: https://link.springer.com/article/10.1007/s10846-021-01383-5.

[9] F. Novák, V. Walter, P. Petráček, T. Báča, and M. Saska, "Fast collective evasion in self-localized swarms of unmanned aerial vehicles," *Bioinspiration & Biomimetics*, vol. 16, no. 6, p. 066 025, Nov. 2021, ISSN: 1748-3190. DOI: 10.1088/1748-3190/ac3060. [Online]. Available: http://dx.doi.org/10.1088/1748-3190/ac3060.

[10] P. Petracek, V. Kratky, M. Petrlik, T. Baca, R. Kratochvil, and M. Saska, "Large-scale exploration of cave environments by unmanned aerial vehicles," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7596–7603, Oct. 2021. DOI: 10.1109/LRA.2021.3098304.

[11] G. Silano, T. Baca, R. Penicka, D. Liuzza, and M. Saska, "Power line inspection tasks with multi-aerial robot systems via signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4169–4176, Apr. 2021. DOI: 10.1109/LRA.2021.3068114.

[12] P. Petráček, V. Walter, T. Báča, and M. Saska, "Bio-inspired compact swarms of unmanned aerial vehicles without communication and external localization," *Bioinspiration &; Biomimetics*, vol. 16, no. 2, p. 026 009, Dec. 2020, ISSN: 1748-3190. DOI: 10.1088/1748-3190/abc6b3. [Online]. Available: http://dx.doi.org/10.1088/1748-3190/abc6b3.

[13] 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. DOI: 10.1109/LRA.2019.2901683.

[14] G. Vásárhelyi, C. Virágh, G. Somorjai, T. Nepusz, A. Eiben, and T. Vicsek, "Optimized flocking of autonomous drones in confined environments," *Science Robotics*, vol. 3, eaat3536, Jul. 2018. DOI: 10.1126/scirobotics.aat3536.

[15] P. Petracek, "Decentralized model of a swarm behavior boids in ros," 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:43708354.

[16] F. Wu, S. Ramchurn, and X. Chen, "Coordinating human-uav teams in disaster response," in *International Joint Conference on Artificial Intelligence (IJCAI-16) (09/07/16 - 15/07/16)*, Apr. 2016, pp. 1–7. [Online]. Available: https://eprints.soton.ac.uk/393725/.

[17] T. Nägeli, C. Conte, A. Domahidi, M. Morari, and O. Hilliges, "Environment-independent formation flight for micro aerial vehicles," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1141–1146. DOI: 10.1109/IROS.2014.6942701.

[18] A. Bürkle, F. Segor, and M. Kollmann, "Towards autonomous micro uav swarms," *Journal of Intelligent and Robotic Systems*, vol. 61, pp. 339–353, Mar. 2011. DOI: 10.1007/s10846-010-9492-x.

[19] S. Hauert *et al.*, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 5015–5020. DOI: 10.1109/IROS.2011.6095129.

[20] M. Quigley *et al.*, "Ros: An open-source robot operating system," vol. 3, Jan. 2009.

[21] R. De Chiara, U. Erra, V. Scarano, and M. Tatafiore, "Massive simulation using gpu of a distributed behavioral model of a flock with obstacle avoidance.," Jan. 2004, pp. 233–240.

[22] C. Reynolds, "Steering behaviors for autonomous characters," Jun. 2002.

[23] 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, no. 6, 1226–1229, Aug. 1995, ISSN: 1079-7114. DOI: 10.1103/physrevlett.75.1226. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.75.1226.

[24] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, 25–34, Aug. 1987, ISSN: 0097-8930. DOI: 10.1145/37402.37406. [Online]. Available: https://doi.org/10.1145/37402.37406.

A Appendix: Attachments

Attachments for this thesis contain the following:

- **ROS Source** - source code of minimalistic ROS node with the swarming model.
- **Deployment Videos** - videos from many angles, including top-down view.

Due to CTU file-size limitations, the quality of videos directly included with this thesis is rather low. High quality deployment videos can be seen on YouTube or downloaded from MRS NAS:

- **YouTube** - quick and easy access, link[1].
- **MRS NAS** - original full quality videos, link[2].

[1] https://www.youtube.com/playlist?list=PL54lWnd1em1foCfidZ01hRsF2nwICWiO9
[2] https://nasmrs.felk.cvut.cz/index.php/s/Pqy0np5Qlff9zz8

B Appendix: AI Software

AI software was used in development of this thesis. More specifically, Large Language Models (LLMs) were used to generate parts of plotting and rosbag processing scripts. AI software was **not** used to write or rephrase any text in this thesis.

- **Ollama**[1] - secure platform for self-hosting LLM models.
- **ChatGPT**[2] - various LLM models provided by OpenAI.

[1] Ollama: https://ollama.com/
[2] ChatGPT: https://chatgpt.com/