Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems

Gesture-based Human Interaction with Relatively-localized Swarms of Unmanned Aerial Vehicles

Bachelor's Thesis

Jan Pražák

Prague, May 2022

Study programme: Open Informatics
Branch of study: Internet of Things

Supervisor: Ing. Jiří Horyna

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Pražák Jan** Personal ID number: **492070**

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

Department / Institute: **Department of Measurement**

Study program: **Open Informatics**

Specialisation: **Internet things**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Gesture-based human interaction with relatively-localized swarms of unmanned aerial vehicles

Bachelor's thesis title in Czech:

Interakce člověka s rojem vzájemně lokalizovaných bezpilotních helikoptér použitím gest

Guidelines:

The aim of this thesis is to design, implement and experimentally verify a method for swarming and interaction of relatively-localized micro aerial vehicles (MAV) with a human using gestures. The thesis will be divided into the following tasks:

To understand model of flocking of ground robots introduced in [1] and the system of MRS group at CTU designed for detection of human gestures.

To design and implement an extension of the flocking model from [1] to make it suitable for MAV deployment.

To design and implement an interaction behavior of the MAV group with a human using gestures detected by onboard cameras.

To compare the designed flocking model with boids flocking model designed for stabilization of MAVs by Multi Robot Systems group at CTU in Prague [2].

To integrate the system into the Robot Operating System (ROS) and verify its behavior with MAV models in realistic Gazebo simulator.

If weather, COVID-19 restrictions, and platforms availability allows, to prepare a HW outdoor experiment

Bibliography / sources:

[1] Shah, Dhruv, and Leena Vachhani. "Swarm aggregation without communication and global positioning." IEEE Robotics and Automation Letters 4.2 (2019): 886-893.
[2] Ahmad, Afzal, et al. "Autonomous aerial swarming in gnss-denied environments with high obstacle density." 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021.
[3] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

Name and workplace of bachelor's thesis supervisor:

Ing. Jiří Horyna Multi-robot Systems FEE

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

Date of bachelor's thesis assignment: **25.01.2022** Deadline for bachelor thesis submission: _____

Assignment valid until:
by the end of summer semester 2022/2023

Ing. Jiří Horyna Supervisor's signature	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

Author statement for undergraduate thesis

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague .....

.....
Jan Pražák

Acknowledgments

I would like to express my gratitude to my supervisor Ing. Jiří Horyna for his guidance and advice on this thesis. My thanks also go to Bc. Afzal Ahmad for his help in setting up the simulation experiments. Further, I want to thank Bc. Yurii Stasinchuk for letting me use his algorithms for position estimation and gesture recognition. My many thanks also go to Isabella Růžičková for her unbiased review of this thesis and to Nicola Trakslová for her continuous support. Lastly, my eternal gratitude belongs to my family, without which this thesis would not have been possible.

Abstract

Autonomous Unmanned Aerial Vehicles (UAVs) and their swarms have been receiving much attention in recent years, both from researchers and the general public. In this thesis, we present a flocking controller suitable for swarms of UAVs derived from a flocking controller used for unicycle Unmanned Ground Vehicles (UGVs). Further, we define gesture-based human-swarm interaction behavior and integrate it with the proposed flocking controller. We then test the emerged system in the realistic Gazebo simulator and compare it to an existing Boids-inspired flocking controller designed by the Multi-robot Systems Group (MRS) at Czech Technical University in Prague (CTU). Lastly, we verify the proposed system in real-world hardware experiments.

Keywords UAV, swarm, interaction, human-swarm interaction

Abstrakt

Autonomní bezpilotní letouny (UAV) a jejich roje získávají v posledních letech mnoho pozornosti, a to jak od výzkumníků, tak od běžné veřejnosti. V této práci představíme rojový kontrolér vhodný pro roje UAV, který je odvozen od rojového kontroléru pro jednokolá bezpilotní pozemní vozidla (UGV). Dále definujeme chování roje při interakci s člověkem za použití gest a integrujeme jej do navrženého rojového kontroléru. Následně vzniklý systém otestujeme v realistickém Gazebo simulátoru a porovnáme jej s existujícím rojovým kontrolérem založeným na Boidech, který byl navržen skupinou Multirobotických systémů (MRS) na Českém vysokém učení technickém v Praze (ČVUT). Nakonec navržený systém ověříme v experimentech s reálným hardwarem.

Klíčová slova UAV, roj, interakce, interakce roje s člověkem

Abbreviations

CTU Czech Technical University

GNSS Global Navigation Satellite System

GPS Global Positioning System

LiDAR Light Detection and Ranging

MRS Multi-robot Systems Group

ROS Robot Operating System

UAV Unmanned Aerial Vehicle

UGV Unmanned Ground Vehicle

RPC Remote Procedure Call

RGBD Red-Green-Blue and Depth

UVDAR UltraViolet Direction and Ranging

PSO Particle Swarm Optimization

MOPSO Multi-Objective Particle Swarm Optimization

LED Light-Emitting Diode

GPM Gravity Points Method

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV) is an aircraft with no pilot on board. Commercial UAVs are typically small quadcopters weighing from a few hundred grams to a few kilograms, including all the carried equipment. UAVs can be either controlled remotely by a human pilot or, as in our case, fly autonomously.

A swarm of UAVs is a group of multiple UAVs working together to achieve a common goal. The UAVs need to be able to actively avoid collisions with one another and coordinate their movements to make reaching the goal more efficient. Exemplary usage of such a swarm can be locating lost or injured people in search and rescue missions, e.g., [3] and [19], as the swarm can cover a larger area than a single UAV could.

In this thesis, we will be dealing with human-swarm interaction. Human-swarm interaction is the execution of indirect commands by a UAV swarm in the presence of a human. Execution of indirect commands means that the swarm performs some actions based on the human's state and position perceived by the swarm itself, for example, using a camera onboard the UAVs, and without any explicit control input from the human such as pressing a button. Examples of such behavior can be a swarm autonomously following the movement of a human or a swarm executing commands according to a human's pose and gestures. Human-swarm interaction is pictured in Fig. 1.1.

In this work, the human can use distinct gestures to instruct the swarm where to fly. Each UAV in the presented swarm system is equipped with an onboard camera. The UAVs use these cameras to perceive the human's gestures, interpret their meaning and fly in the requested direction.

We solved the tackled problem without explicit inter-agent communication. Communication is used to share GNSS coordinates of UAVs when GNSS is employed for mutual localization of the UAVs. Furthermore, an implicit communication strategy is utilized for gesture consensus among the UAVs.

1.1 State of the art

One of the two main goals of this thesis is to create a flocking controller suitable for UAVs based on the flocking controller described in [4]. In [4], an agent is considered to be a unicycle UGV. Each agent can sense only the closest other agent in a cone in front of them. The only actions an agent can perform are moving in the direction of their heading and rotating to change their heading.

A traditional approach to controlling a swarm is using a Boids controller ([1], [15], [17], [22]), which was derived in [22] by observing the behavior of flocks, herds, and schools. A Boids controller is based on combining three virtual forces, also called vectors. These virtual

Figure 1.1: Example of human-swarm interaction. The swarm consists of three UAVs, highlighted by red circles, the human interacting with them is highlighted by a yellow rectangle.

forces are the proximal force that keeps the swarm together, the navigation force that moves the swarm toward the goal, and the collision force responsible for collision avoidance.

In [1], the Boids-controlled swarm uses UltraViolet Direction and Ranging (UVDAR), see [5]–[7], for relative localization of UAVs, and Light Detection and Ranging (LiDAR) for localization of other obstacles. The swarm was deployed in a dense natural forest to demonstrate its behavior in an environment where Global Navigation Satellite System (GNSS) might be unreliable or even entirely unavailable.

Another popular bio-inspired flocking algorithm called BEECLUST is presented in [11]. The algorithm was derived by observing the behavior of honeybees. BEECLUST is a relatively simple and computationally inexpensive yet robust and powerful flocking algorithm that only uses commands to move forward, turn away, and stay.

An even simpler algorithm called Particle Swarm Optimization (PSO) that relies on a single equation is proposed in [21]. Thanks to its simplicity and versatility, PSO has become very widely used ([16], [20]). The popularity of PSO led to the creation of its variations. One of those variations is Multi-Objective Particle Swarm Optimization (MOPSO), described in [18], which was used in path planning for car-like robots [8]. A local version of PSO in which the agents communicate only with other agents in their local neighborhood is used in [14] to detect radiation sources.

A more high-level approach to designing swarm controllers is presented in [15]. The authors use a sample/interpolate method to map low-level controller rules to abstract properties that are easier to imagine for a human. Then the authors demonstrate this technique by configuring a Boids-controlled swarm's desired area density and letting the proposed algorithm find the required parameters for the low-level virtual forces of the Boids.

The other goal of this thesis is to design an interaction behavior between the UAV swarm and a human operator. Gesture-based human-swarm interaction was also used in [9] for selecting individual UAVs or groups of UAVs. The human operator was equipped with colored gloves and a vest to make observing the human and their gestures easier for the UAVs. The UAVs used face detection for relative localization both among themselves and with the human.

The authors of [10] discuss different types of human-swarm interaction, which they call intermittent and environmental. The control methods available to the human for each type of interaction are called *beacon* control and *selection* control, respectively. Both types can have the same effect on a particular robot in the swarm, but they differ in how the affected robots are selected. *Selection* control allows the human to select individual robots for interaction. Thus, *selection* control is temporally persistent, meaning the robots remain selected until the human deselects them. On the other hand, *beacon* control enables the human to place beacons in the environment that influence nearby robots. Therefore, *beacon* control is spatially persistent, meaning the robots are influenced for as long as they remain in the affected area. The authors then compare the two control methods in terms of their effectiveness in human-swarm interaction.

In [12], the swarm uses the PSO algorithm mentioned above for finding various optima of a fitness function in an environment. The authors then combine the PSO algorithm with their original method called Gravity Points Method (GPM), which is used for representing goals. The goals in GPM can be either virtual attractive points, which have a positive effect on the fitness function, or virtual repulsive points, which have a negative effect on the fitness function. A human operator can then place these virtual points according to their knowledge of the searched environment and thus help the swarm find the optima faster.

Lastly, an interesting approach to human-swarm interaction is presented in [14]. There, a human operator takes direct control over one or more agents in the swarm and uses them to guide the other agents. The system was tested in a radiation source localization scenario. The agents initially move in random directions, and the human operator can change the state of selected agents so that they instruct others to move in a particular direction. Once a radiation source is found, the agents converge toward it using a local variation of PSO.

1.2 Problem statement

The goal of this thesis was to design and implement an interaction behavior of a UAV swarm with a human using gestures detected by onboard cameras. For this purpose, a design and implementation of an extension of the flocking controller from [4] suitable for UAVs was to be derived.

The UAV swarm should not endanger the human operator. Apart from this, the UAV swarm would be operating in an obstacle-free environment with only other UAVs taken into account for collision avoidance.

No explicit inter-agent communication is used for the purposes of the flocking controller.

We assume that only a single human can be in sight of the UAVs in the swarm.

1.3 Mathematical notation

Mathematical notation used in this thesis is defined in Table 1.1.

$\lvert x \rvert$	absolute value
$\mathbf{x}$	vector
$x = \mathbf{a}^\intercal \mathbf{b}$	inner product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\lVert \mathbf{x} \rVert = \sqrt{\mathbf{x}^\intercal \mathbf{x}}$	Euclidean norm
$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\lVert \mathbf{x} \rVert}$	unit vector
$\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$	elements of the *standard basis*
$\mathbf{x}_{(n)} = \mathbf{x}^\intercal \hat{\mathbf{e}}_n$	n^{th} vector element (row), $\mathbf{x}, \mathbf{e} \in \mathbb{R}^3$
$\dot{x}, \ddot{x}, \dddot{x}, \ddddot{x}$	1st, 2nd, 3rd, and 4th time derivative of x

Table 1.1: Mathematical notation, nomenclature and notable symbols.

Chapter 2

Preliminaries

2.1 UGV flocking controller

The flocking controller used in this thesis is based on the flocking controller deployed in UGV in [4]. There, the agents are assumed to be unicycle vehicles. These vehicles have a car-like movement as they can only go forward and turn to the left and to the right. No inter-agent communication was used. Each agent can only sense the closest agent in a circular sector pointed in the direction of their movement. All the agents are aware of a common target, and their goal is to reach this target.

The high-level controls of the UGVs are divided into two subsystems - the *free subsystem* and the *engaged subsystem*. In the *free subsystem*, the agent has detected no other agent, accelerates until it reaches its maximum velocity, and turns toward the target. In the *engaged subsystem*, the agent has detected another agent, decelerates, and turns away from the detected agent to avoid a collision. No explicit cohesion is enforced. The cohesion comes implicitly from the fact that the swarm was initially grouped together and that all the agents try to reach the same target.

2.2 Robot Operating System

Robot Operating System (ROS)[1] [13] is a set of tools and libraries designed for developing robot applications. Within ROS, individual processes are referred to as nodes. These nodes can communicate with one another using topics and services. Individual topics are uniquely identified by their names. A single topic can be thought of as the classic producer-consumer scenario with an arbitrary number of both producers and consumers. Producer nodes publish messages onto a given topic, and the subscribed consumer nodes receive them. A single node can be both a producer and a consumer for a given topic. An exemplary communication using ROS topics can be seen in Fig. 2.1. A service is similar to a Remote Procedure Call (RPC). A node provides an RPC which can be invoked by other nodes using its name. After the RPC has been executed, a response is returned to the caller. An example of a ROS service call is presented in Fig. 2.2.

2.3 Gazebo simulator

Gazebo simulator[2] (shown in Fig. 2.3) is a free and open-source simulator focused on applications in robotics. Its realistic physics and environment allow us to verify our algorithms

[1] `https://www.ros.org/`
[2] `http://gazebosim.org/`

reliably without having to conduct real-world experiments and endangering our hardware by running untested software.

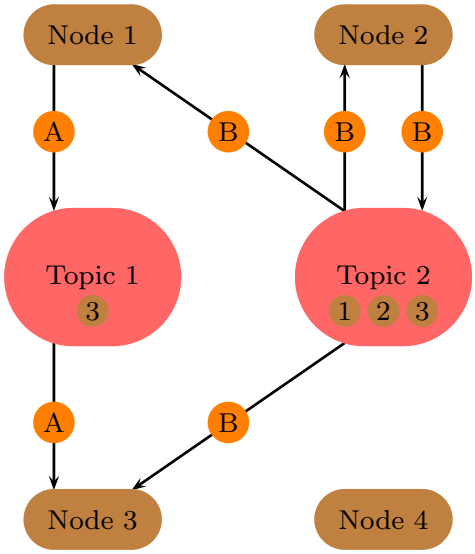


Figure 2.1: A flow of ROS topics. Node 1 publishes message A to Topic 1. Only Node 3 is subscribed to Topic 1, therefore only Node 3 receives the message A. Node 2 publishes message B to Topic 2. Nodes 1, 2, and 3 are subscribed to Topic 2; therefore, Nodes 1, 2, and 3 receive the message B. Node 4 is not subscribed to any topic, therefore it does not receive any of the messages.

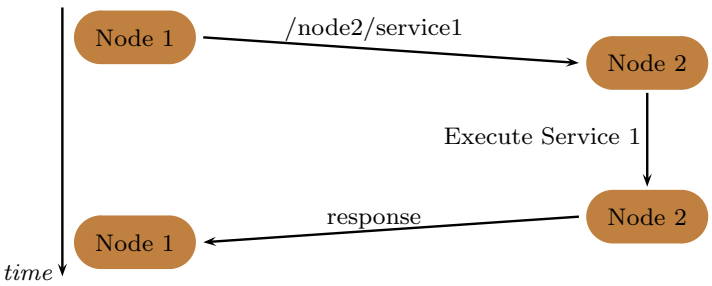


Figure 2.2: A flow of ROS services. Node 1 invokes Service 1 provided by Node 2. Node 2 executes the service, then returns a response to Node 1.

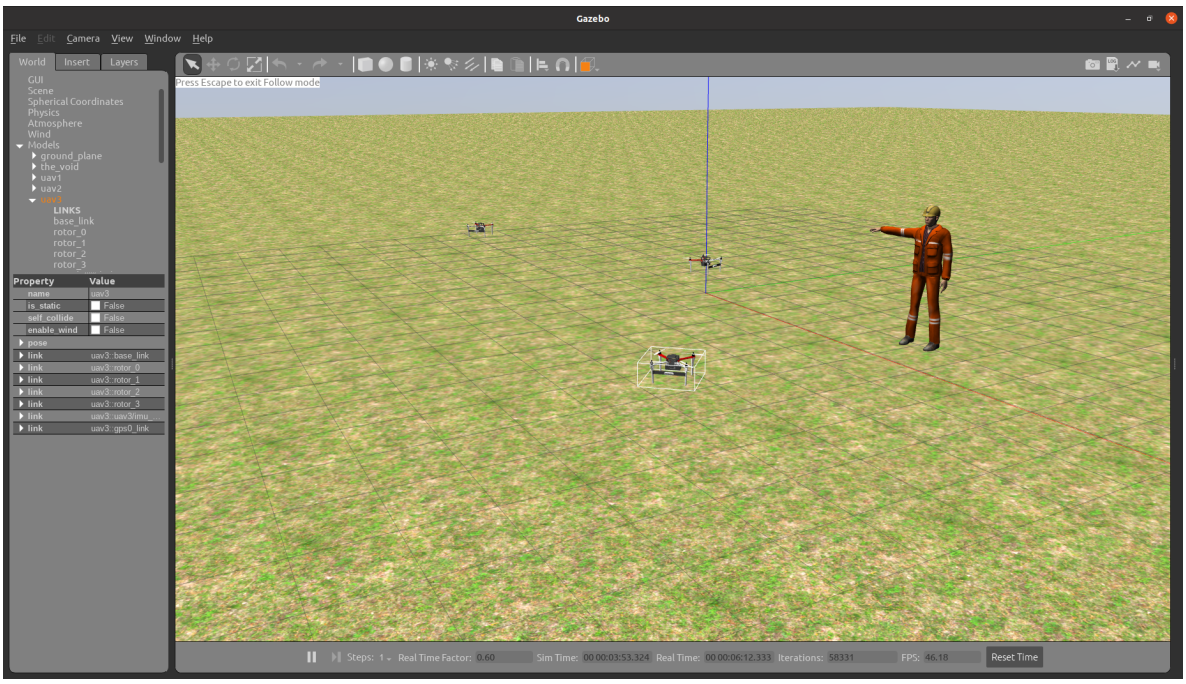


Figure 2.3: Screenshot of the Gazebo simulator. Three UAVs and a human are present in the simulation.

Chapter 3

Flocking controller

In this chapter, we describe the extension of the flocking controller for UGVs presented in [4] that suits the requirements of UAVs.

The UGVs in [4] are unicycle vehicles that can only go forward or turn to the left or to the right. However, UAVs can always go in any direction no matter their heading. Therefore, in this work, we distinguish two kinds of headings of a UAV - real heading, i.e., the yaw angle of the UAV, and virtual heading, i.e., the direction of movement of the UAV. Additionally, unlike UGVs, UAVs can move along the vertical axis. The controller described in this thesis does not take advantage of the vertical axis, and thus the UAVs are controlled in a 2D plane. A constant altitude is maintained by the lower-level controls of the UAV [2].

3.1 UAV kinematics

The position $\mathbf{x}_i$ of the i-th agent in the Euclidean space with axes $\hat{\mathbf{e}}_1$, $\hat{\mathbf{e}}_2$, $\hat{\mathbf{e}}_3$ is defined by its coordinates x_i, y_i, and z_i. Each agent is able to measure its virtual heading θ_i and its linear velocity v_i. The virtual heading θ_i is the angle between the vector of the UAV's lateral velocity and $\hat{\mathbf{e}}_1$. Further, we define the agent's maximum linear velocity v_{imax}, and safe distance s_i. The safe distance defines a radius around the UAV in which no other UAV should be located to avoid collisions. The UAV kinematics is depicted in Fig. 3.1.

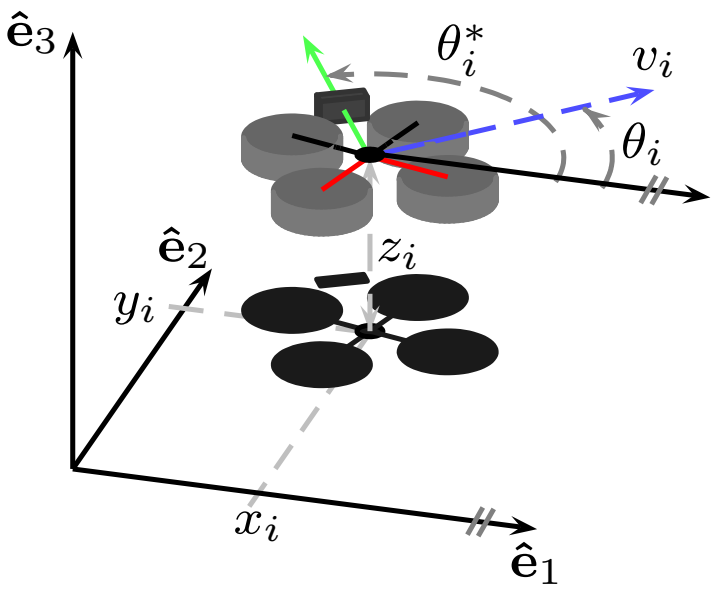


Figure 3.1: Depiction of the UAV kinematics.

3.2 UAV sensing

For the purposes of collision avoidance, the agents need to be able to detect other agents in their vicinity. Each agent i has a virtual spherical sector pointed in the agent i's direction of travel. The agent i only detects the closest other agent within this sector, whom the agent i then actively avoids. Detecting only within a narrow field of view is advantageous because it limits the number of agents that need to be concerned. Additionally, such behavior is common in nature, for example, in bees who only adjust their flight based on the bees (and other obstacles) in front of them, which also inspired the BEECLUST swarm algorithm described in [11].

3.2.1 Detection spherical sector

We are working in a three-dimensional space, so instead of a detection cone (or rather a circular sector) as described in [4], we use a detection spherical sector. Each agent can detect the closest other agent within its detection spherical sector. For the agent i, such a spherical sector is defined by a detection range d_i and a detection angle φ_i. The spherical sector is headed in the direction θ_i. A depiction of UAVs and their detection spherical sectors is in Fig. 3.2.

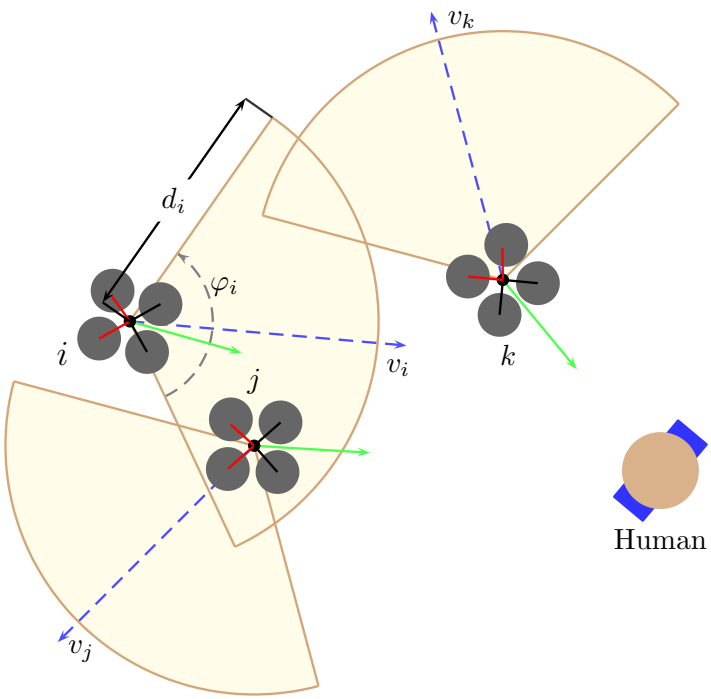


Figure 3.2: Top-down depiction of UAVs. Dashed blue lines denote a UAV's velocity. Detection spherical sectors are shown in yellow. UAV i detects UAV j, no other UAVs are detected. Note that the real heading of the UAVs (in green) is toward the human rather than in the direction of travel.

3.2.2 UAV detection

Let us define the vector determining the relative position between the UAVs i and j

$$\mathbf{x}_{ij} := \mathbf{x}_j - \mathbf{x}_i.$$

Further, we transform $\mathbf{x}_{ij}$ to polar coordinates:

$$\begin{aligned} \mathbf{x}_{ij(1)} &= \rho_{ij} \sin \vartheta_{ij} \cos \phi_{ij}, \\ \mathbf{x}_{ij(2)} &= \rho_{ij} \sin \vartheta_{ij} \sin \phi_{ij}, \\ \mathbf{x}_{ij(3)} &= \rho_{ij} \cos \vartheta_{ij}. \end{aligned} \tag{3.1}$$

From (3.1), we extract the values of the polar coordinates:

$$\begin{aligned} \rho_{ij} &= \|\mathbf{x}_{ij}\|, \\ \vartheta_{ij} &= \arccos\left(\frac{\mathbf{x}_{ij(3)}}{\rho_{ij}}\right), \\ \phi_{ij} &= \begin{cases} \arccos\left(\frac{\mathbf{x}_{ij(1)}}{\rho_{ij} \sin \vartheta_{ij}}\right) & \text{if } \mathbf{x}_{ij(2)} \geq 0, \\ 2\pi - \arccos\left(\frac{\mathbf{x}_{ij(1)}}{\rho_{ij} \sin \vartheta_{ij}}\right) & \text{otherwise.} \end{cases} \end{aligned} \tag{3.2}$$

UAV j is in the i-th UAV's detection range if

$$\rho_{ij} \leq d_i. \tag{3.3}$$

Then, the inequality

$$\theta_i - \frac{\varphi_i}{2} \leq \phi_{ij} \leq \theta_i + \frac{\varphi_i}{2} \tag{3.4}$$

checks if agent j is horizontally inside the detection spherical sector of agent i. Finally, the inequality

$$\frac{\pi}{2} - \frac{\varphi_i}{2} \leq \vartheta_{ij} \leq \frac{\pi}{2} + \frac{\varphi_i}{2} \tag{3.5}$$

holds true only if agent j is vertically inside the detection spherical sector of agent i.

If agent j is the closest agent to agent i for whom all of the inequalities (3.3), (3.4), (3.5) are satisfied, agent i detects agent j.

3.3 Controls

At a given moment, an agent can be in exactly one of two subsystems, called the *free subsystem* and the *engaged subsystem*. When in the *free subsystem*, the agent always flies toward the target unless already there. In the *engaged subsystem*, agent i detects another agent j. Agent i decelerates and turns away from agent j so that the agents do not collide. Once agent i no longer has agent j in the detection spherical sector, agent i transitions to the *free subsystem* and starts accelerating and turning toward the target. A depiction of the controls is in Fig. 3.3.

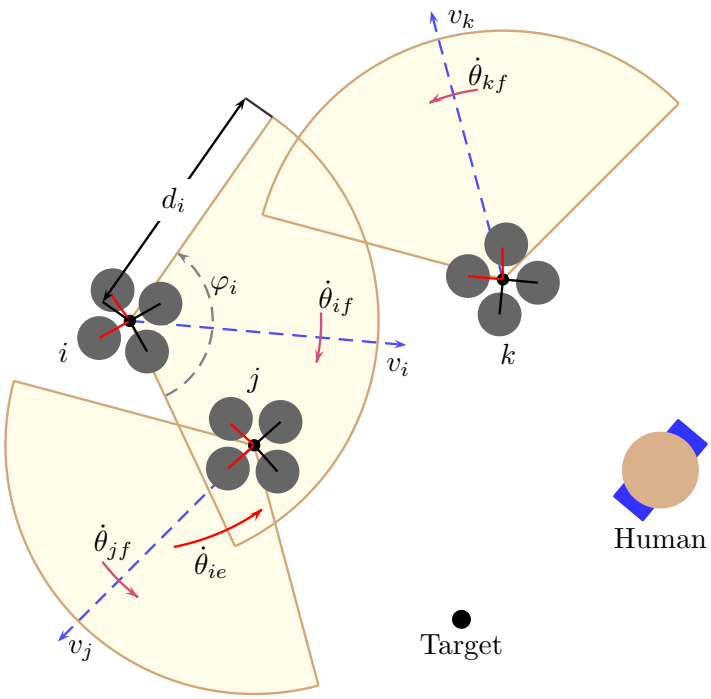


Figure 3.3: Top-down depiction of UAVs. The purple arrows show the virtual force turning the agents toward the target. Agent i detects agent j and enters the *engaged subsystem*, the red arrow $\dot{\theta}_{ie}$ depicts the virtual force turning agent i away from agent j to avoid a collision.

3.3.1 Free subsystem

Assume that agent i is aware of a circular target area $T(i)$ with coordinates $T_x(i)$, $T_y(i)$, and radius $T_r(i)$. Let us define the vector from the i-th UAV's position to the target $T(i)$'s center

$$\mathbf{d}_{iT} := \begin{bmatrix} T_x(i) \\ T_y(i) \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

Further, we calculate the required virtual heading φ_{iT} from agent i to target $T(i)$

$$\varphi_{iT} = \begin{cases} \arccos \widehat{\mathbf{d}}_{iT}(1) & \text{if } \widehat{\mathbf{d}}_{iT}(2) \geq 0, \\ 2\pi - \arccos \widehat{\mathbf{d}}_{iT}(1) & \text{otherwise.} \end{cases} \tag{3.6}$$

From that, we get the virtual heading rate using which the agent tries to head toward the target:

$$\dot{\theta}_i = \frac{\varphi_{iT} - \theta_i}{\pi} k_f, \tag{3.7}$$

where k_f is a free parameter satisfying

$$k_f > \frac{v_{imax}}{T_r(i)},$$

and the value of the difference $(\varphi_{iT} - \theta_i)$ is in the range $[-\pi, \pi)$. As the UAV's virtual heading θ_i approaches the heading to target φ_{iT}, the virtual heading rate $\dot{\theta}_i$ linearly approaches zero, reducing oscillation of θ_i around φ_{iT}.

The UAV's desired velocity is calculated using the formula from [4]

$$v_i = v_{imax}(1 - \frac{v_{imax} - v_i^{t^i_{f_e}}}{v_{imax}} e^{-t^i_f}), \tag{3.8}$$

where $v_i^{t^i_{f_e}}$ is the UAV's velocity when it last entered the *free subsystem* and t^i_f is the time spent in the *free subsystem* measured since the subsystem was last entered. Apart from this velocity calculation, we also want to stop the UAV once it reaches the target area. Therefore, constant deceleration

$$a_i^* := \frac{v_{imax}^2}{2T_r(i)}, \tag{3.9}$$

is applied as long as the UAV is in the target area, its velocity v_i is greater than zero, and it is in the *free subsystem*. The UAV is inside the target area when the inequality

$$(x_i - T_x(i))^2 + (y_i - T_y(i))^2 \leq (T_r(i))^2 \tag{3.10}$$

is satisfied.

The rules defined for the *free subsystem* ensure that the agent turns its virtual heading toward the target and accelerates until it reaches the maximum linear velocity v_{imax}, unless in the target area where the agent gradually slows down until it stops. If the agent is not aware of any target, it retains its current virtual heading and sets its velocity to zero.

3.3.2 Engaged subsystem

In the *engaged subsystem*, agent i has detected agent j and estimates agent j's position $\mathbf{x}_j$. If more than one agent is inside agent i's detection spherical sector, only the closest one is concerned. While in the *engaged subsystem*, agent i steers away from the closest agent in its path and reduces its linear velocity v_i, effectively avoiding other agents in the swarm while trying to head toward the target.

We calculate the virtual heading rate in the *engaged subsystem* by subtracting (or adding, based on $sgn(\delta_{ij} - \theta_i)$) a constant from the value calculated in (3.7):

$$\dot{\theta}_i = \frac{\varphi_{iT} - \theta_i}{\pi} k_f - k_e sgn(\delta_{ij} - \theta_i), \tag{3.11}$$

where δ_{ij} is the angle between $\mathbf{p}_z(x_{ij})$ and $\hat{\mathbf{e}}_1$, k_e is a free parameter satisfying

$$k_e > \frac{v_{imax}}{T_r(i)} + 2\frac{v_{max}}{s_i},$$

and v_{max} is the maximum linear velocity among all the agents:

$$v_{max} := max_k(v_{kmax}). \tag{3.12}$$

The desired velocity in the *engaged subsystem* is given by the formula from [4]

$$v_i = max\{v_i^{t^i_{e_e}} - t^i_e \lambda_i, 0\}, \tag{3.13}$$

where $v_i^{t^i_{e_e}}$ is the linear velocity v_i of agent i when it last entered the *engaged subsystem*, t^i_e is the time spent in the *engaged subsystem* measured since the subsystem was last entered, and λ_i is a free parameter satisfying

$$\lambda_i > \frac{v_{max}^2}{d_i - s_i}. \tag{3.14}$$

3.3.3 Direct change of virtual heading

Assume that agent i does not have any other agents in its vicinity. Then, assume that agent i's target $T(i)$ changes and is now in the opposite direction from agent i than agent i's virtual heading θ_i. Agent i would then move in the direction θ_i, i.e., *away* from the target, until the agent has turned around using the virtual heading rate $\dot{\theta}_i$ from (3.7).

It is apparent that, in such a case, the agent could start moving immediately toward the target. This can be described as follows:

Whenever target $T(i)$ of agent i changes and agent i is in the *free subsystem*, agent i can immediately set its virtual heading θ_i toward the target unless it would cause agent i to leave the *free subsystem*.

3.3.4 Resulting behavior

In the *free subsystem*, the agent accelerates and turns toward the target; thus, the agent approaches the target. In the *engaged subsystem*, agent i has detected agent j. If agent j is not moving, agent i's virtual force from (3.7) keeps competing with (3.11) until agent i has circled agent j. At this point, agent i can no longer detect agent j and therefore stays in the *free subsystem* and continues toward the target. If agent j is moving, it should be moving in generally the same direction as agent i because the agents should have a similar target. If agent j has a low velocity (e.g., because another agent blocks agent j or because agent j has reached its target), agent i still attempts to circle agent j as already described. Otherwise, the frequent deceleration of agent i as it repeatedly enters the *engaged subsystem* means that agent j travels at a higher velocity than agent i. Thus, agent j eventually flies out of agent i's detection range, at which point agent i stays in the *free subsystem* and continues toward the target.

Chapter 4

Human-swarm interaction

This chapter will extend the flocking controller presented in Chapter 3 to include human-swarm interactions.

The goal is for the agents to execute commands in the form of traveling in a given direction, i.e., going to the left, right, forward, or backward. Which command should be executed (if any) is decided by a human operator. The human performs a gesture, such as raising a hand. The agents detect the gesture using their onboard RGBD camera, evaluate the gesture's meaning in the context of their own position and the human's rotation, and execute the command. For that, the agents need to observe the human using the RGBD cameras. It is improbable that all the agents would be able to see the human to perceive the gesture. For example, there could be an obstacle in the form of another agent, the human could be seen from an inappropriate angle, or the human could be too far away. Because the flocking controller does not provide any explicit cohesion, we must take these problems into account and make sure that all the agents agree on the perceived gesture and have a similar estimate of the human's rotation so that we can achieve a uniform execution of commands and keep the swarm grouped. Therefore, we use inter-agent communication and a consensus algorithm. The interpreted gestures are shown in Fig. 4.1.

4.1 Extending the flocking controller

Agent i uses an RGBD camera to estimate four parameters of the human H - the $\hat{\mathbf{e}}_1$ coordinate $H_x(i)$, the $\hat{\mathbf{e}}_2$ coordinate $H_y(i)$, the yaw angle $H_r(i)$, and the current gesture $H_G(i)$. The yaw angle and the gesture are used to determine the reaction of the swarm to the gesture. Additionally, each agent measures its own real heading θ_i^*.

4.2 Observing the human

Each agent is equipped with a single RGBD camera pointed in the direction of the UAV's real heading θ_i^*. The agents use these cameras to measure the human's position and yaw angle and to estimate the human's gesture. For that, the agents need to turn themselves (and thus their RGBD cameras) toward the human.

For simplicity reasons, the agents determine their desired real heading rate $\dot{\theta}_i^*$ similarly to their virtual heading rate in (3.7). First, let us define the relative vector from agent i's position toward the human H

$$\mathbf{d}_{iH} := \begin{bmatrix} H_x(i) \\ H_y(i) \end{bmatrix} - \begin{bmatrix} x_i \\ y_i \end{bmatrix}.$$

Then, we can calculate the required heading φ_{iH} from agent i to human H:

$$\varphi_{iH} = \begin{cases} \arccos \widehat{\mathbf{d}}_{iH(1)} & \text{if } \widehat{\mathbf{d}}_{iH(2)} \geq 0, \\ 2\pi - \arccos \widehat{\mathbf{d}}_{iH(1)} & \text{otherwise.} \end{cases} \tag{4.1}$$

From that, we get the desired real heading rate using which the agent tries to head toward the human:

$$\dot{\theta}_i^* = \frac{\varphi_{iH} - \theta_i^*}{\pi} k_h, \tag{4.2}$$

where k_h is a free parameter satisfying

$$k_h > \frac{v_{imax}}{2},$$

and the value of the difference $(\varphi_{iH} - \theta_i^*)$ is in the range $[-\pi, \pi)$. As the UAV's real heading θ_i^* approaches the heading to human φ_{iH}, the real heading rate $\dot{\theta}_i^*$ approaches zero, reducing oscillation of θ_i^* around φ_{iH}.

Using the formula (4.2), the agents turn their front toward the human, which allows them to monitor the human's gestures and execute commands. If the agents are not aware of any human, they retain their current real headings and do not execute any commands.

4.3 Position estimation

We assume there is at most one human in sight of the agents. The agents measure the human's position using their onboard cameras and exchange the information with other agents. This communication is to prevent a situation when an agent's real heading is turned away from the human, and thus the agent cannot observe the human's position and gesture.

Each agent j sends perceived information about the human to all agents (including itself). The information contains the human's coordinates $H_x^*(j)$ and $H_y^*(j)$ and the human's yaw angle $H_r^*(j)$. Having received this information, agent i updates its values of the human's coordinates

$$H_x(i) = \frac{H_{x_{old}}(i) + H_x^*(j)}{2}, \tag{4.3}$$

$$H_y = \frac{H_{y_{old}}(i) + H_y^*(j)}{2}, \tag{4.4}$$

where $H_{x_{old}}(i)$ and $H_{y_{old}}(i)$ are the previous values of $H_x(i)$ and $H_y(i)$, respectively. Agent i also updates the value of the human's yaw angle according to the formula for calculating a circular mean of two angles:

$$\begin{aligned} H_{r_x} &= \cos\left(H_{r_{old}}(i)\right) + \cos\left(H_r^*(j)\right), \\ H_{r_y} &= \sin\left(H_{r_{old}}(i)\right) + \sin\left(H_r^*(j)\right), \\ H_r &= \operatorname{atan2}\left(H_{r_y}, H_{r_x}\right), \end{aligned} \tag{4.5}$$

where $H_{r_{old}}(i)$ is the previous value of $H_r(i)$.

4.4 Gesture consensus

All the UAVs must execute the same command simultaneously for the swarm to stick together. Therefore, a consensus on the perceived gesture must be reached. We use a decentralized voting-based consensus algorithm.

Whenever a UAV j registers a gesture $H_G(j)$ from the human H, it sends a message M_{jG} to all UAVs in the swarm (including itself).

When a UAV i receives the message M_{jG}, it first checks if agent j is marked that it has already voted. If not, it marks that UAV j has voted, increments the number of votes for gesture $H_G(j)$ by one, and starts a timer t_{iC}. If the timer t_{iC} is already running, it is restarted instead.

When the timer t_{iC} expires, the gesture with the most votes is taken as the valid one. If there are more of those, then, for each of these gestures, we find the UAV with the highest identifier that voted for the given gesture and take the gesture with the lowest such identifier. Alg. 1 shows the described algorithm used after the timer t_{iC} expires. Examples of runs of the consensus algorithm can be found in Table 4.1.

Algorithm 1 Gesture consensus algorithm

```
procedure GestureConsensus(votes)                    ▷ Map⟨Integer, Gesture⟩ votes
    SetZeroes(gestureCounts)                         ▷ Map⟨Gesture, Integer⟩ gestureCounts
    determinedGesture ← None
    maxVotes ← 0
    for uav ∈ UAVs do                                ▷ UAVs are sorted by their id in ascending order
        gesture ← votes[uav.id]
        if gesture ≠ None then                       ▷ gesture = None means UAV did not vote
            gestureCounts[gesture] ← gestureCounts[gesture] + 1
            if gestureCounts[gesture] > maxVotes then
                determinedGesture ← gesture
                maxVotes ← gestureCounts[gesture]
            end if
        end if
    end for
    return determinedGesture
end procedure
```

UAV identifier	Votes			
	Ex. 1	Ex. 2	Ex. 3	Ex. 4
1	Left	-	Forward	Forward
2	Right	Forward	Left	Left
3	Backward	-	Right	Right
4	-	-	Right	Left
5	Left	-	Left	Right
Result	Left	Forward	Right	Left

Table 4.1: Consensus algorithm examples.

The solution expects that an arbitrary number of agents can break down and stop

communicating. In that case, the damaged agents would neither vote for gestures nor send their perception of the human's position. The remaining agents can continue to function without any noticeable difficulties apart from a possible decrease in precision in the human's position and gesture estimation.

4.5 Command execution

After the swarm has agreed on the perceived gesture H_G, each agent i needs to calculate the coordinates of its target

$$T(i) = \begin{bmatrix} x_i + d_f \cdot \cos(H_r(i) + \delta) \\ y_i + d_f \cdot \sin(H_r(i) + \delta) \end{bmatrix},$$

where d_f is the distance the UAV should fly when executing one command and

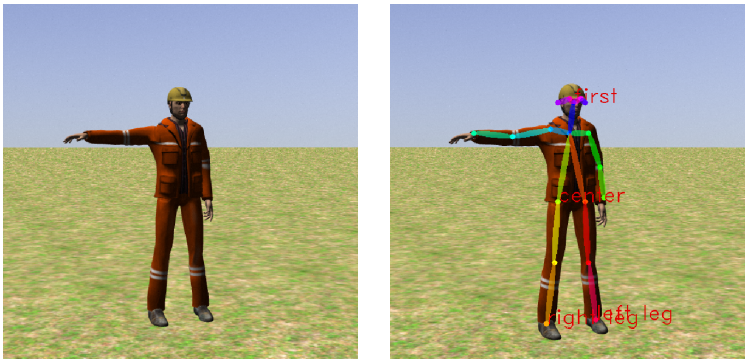
$$\delta = \begin{cases} -\frac{\pi}{2} & \text{if } H_G \text{ is "Left"}, \\ \frac{\pi}{2} & \text{if } H_G \text{ is "Right"}, \\ \pi & \text{if } H_G \text{ is "Forward"}, \\ 0 & \text{if } H_G \text{ is "Backward"}, \end{cases} \tag{4.6}$$

Then, the UAV flies toward the target, as described in Chapter 3.

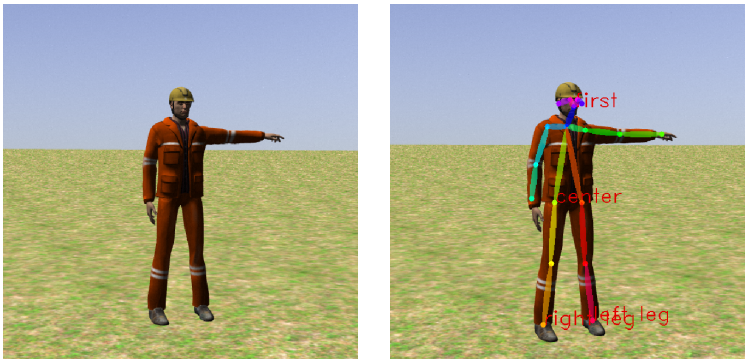
The UAV can receive a new command before finishing the previous one, which results in it immediately starting to execute the new command and therefore not flying the entire distance d_f when executing the previous command. An example of target coordinates calculations is shown in Fig. 4.2.

4.6 Avoiding the human

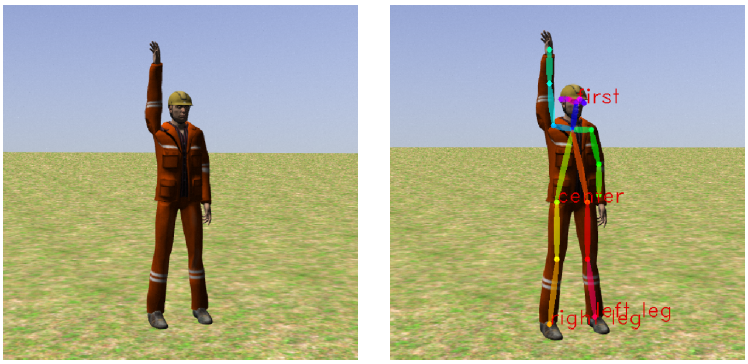
The swarm should not pose a danger to the human operating it. Therefore, a virtual UAV is located at the human's position. The UAVs are thus effectively avoiding collisions with the human according to the same rules applied for avoiding collisions among themselves.



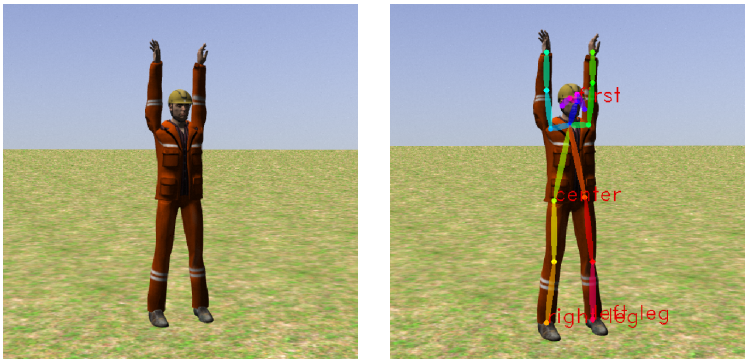
(a) Left gesture.



(b) Right gesture.



(c) Forward gesture.



(d) Backward gesture.

Figure 4.1: Gestures recognized by the gesture estimator. The images on the left-hand side show raw images from a UAV's RGBD camera. The images on the right-hand side show intermediate output from the gesture estimator depicting found keypoints and connections.

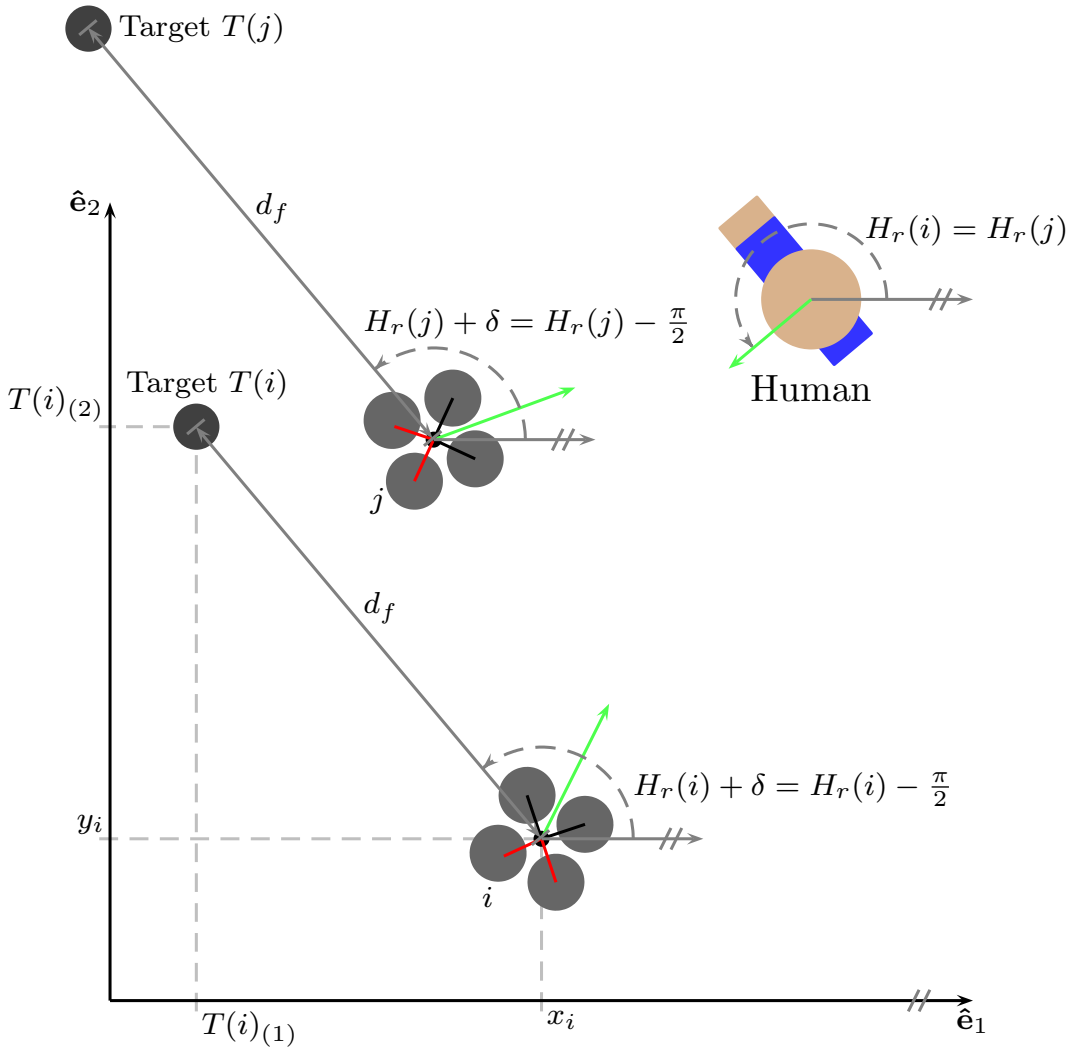


Figure 4.2: Top-down depiction of target coordinates calculations for agents i and j. $H_r(i)$ is equal to $H_r(j)$ for a better visual representation. The detected gesture is 'Left'; therefore, $\delta = -\frac{\pi}{2}$.

Chapter 5

Simulation

This chapter presents the results from testing the proposed flocking controller and human-swarm interaction in the realistic Gazebo simulator. Additionally, some tests utilize the novel UVDAR technology, see [5]–[7], for relative localization of other UAVs instead of sharing GNSS coordinates via implicit communication. Using the UVDAR system instead of sharing GNSS coordinates can be advantageous in situations where communication between the UAVs may not be possible, for example, due to high levels of interference.

The realistic Gazebo simulator allows us to test the swarm's behavior without endangering real hardware or living beings. Additionally, our resources are not limited by the number of available UAVs or other hardware but only by the performance of the computer running the simulation. Although the conditions in the simulation are not identical to those in the real world, the results from the simulation can be strong indicators of a swarm's performance and can reliably verify the correctness of proposed algorithms.

The tests were performed with parameters set as shown in Table 5.1.

Parameter	Symbol	Value
Maximum linear velocity	v_{imax}	$3.0\,\mathrm{m\,s^{-1}}$
Safe distance	s_i	differs per test
Detection range	d_i	differs per test
Detection angle	φ_i	$1.27 \approx \frac{\pi}{2} - 0.3$
Target radius	$T_r(i)$	$8.0\,\mathrm{m}$
Free subsystem virtual heading rate factor	k_f	3.0
Engaged subsystem virtual heading rate factor	k_e	6.2
Engaged subsystem deceleration factor	λ_i	$3.2\,\mathrm{m\,s^{-2}}$
Real heading rate factor	k_h	3.0
Gesture consensus timer	t_{iC}	$1.0\,\mathrm{s}$
Distance to fly	d_f	$20.0\,\mathrm{m}$

Table 5.1: Parameter settings used for experiments run in the Gazebo simulator.

5.1 GNSS sharing, shorter detection range

In the first test, we placed 6 UAVs in a grid with two rows and three columns. Individual rows and columns were spaced 5 meters apart. Each UAV's detection range d_i was set to 5 meters and its safe distance s_i to 2 meters. The agents used implicit communication to share their GNSS coordinates among themselves. We then set each UAV's target $T(i)$ 20 meters away from UAV i along the $\hat{\mathbf{e}}_\mathbf{2}$ axis. After a few seconds, we changed the targets to be 20 meters away from UAV i along the $\hat{\mathbf{e}}_\mathbf{1}$ axis. After a few more seconds, we changed the targets

for the final time to be 20 meters away from UAV i in the opposite direction than the $\hat{\mathbf{e}}_2$ axis. The sequence of changing targets should mimic a human headed in the direction of the $\hat{\mathbf{e}}_2$ axis and showing the gestures Backward, Right, and Forward. The resulting trajectories are shown in Fig. 5.1.

From Fig. 5.1, we can see that the UAVs maintained their position within the swarm well, with only UAV 2 ending closer to UAV 1 than it was at the start, as shown in Fig. 5.2. UAV 2 detected UAV 1 about 10 seconds into the flight and had to start avoiding it, which consequently caused UAV 2 to be detected by UAV 5. This led to UAV 5 ending farther from the other UAVs, as shown in Fig. 5.2. From Fig. 5.2, we can further see that no two UAVs had breached one another's safe distance s_i of 2 meters. In Fig. 5.3, we can see the velocities of the UAVs. The UAVs successfully maintained the maximum linear velocity v_{imax} of $3\,\mathrm{m\,s^{-1}}$, with deviations only when another UAV was detected, which is expected, or when turning, which is probably caused by the low-level controls of the UAVs.

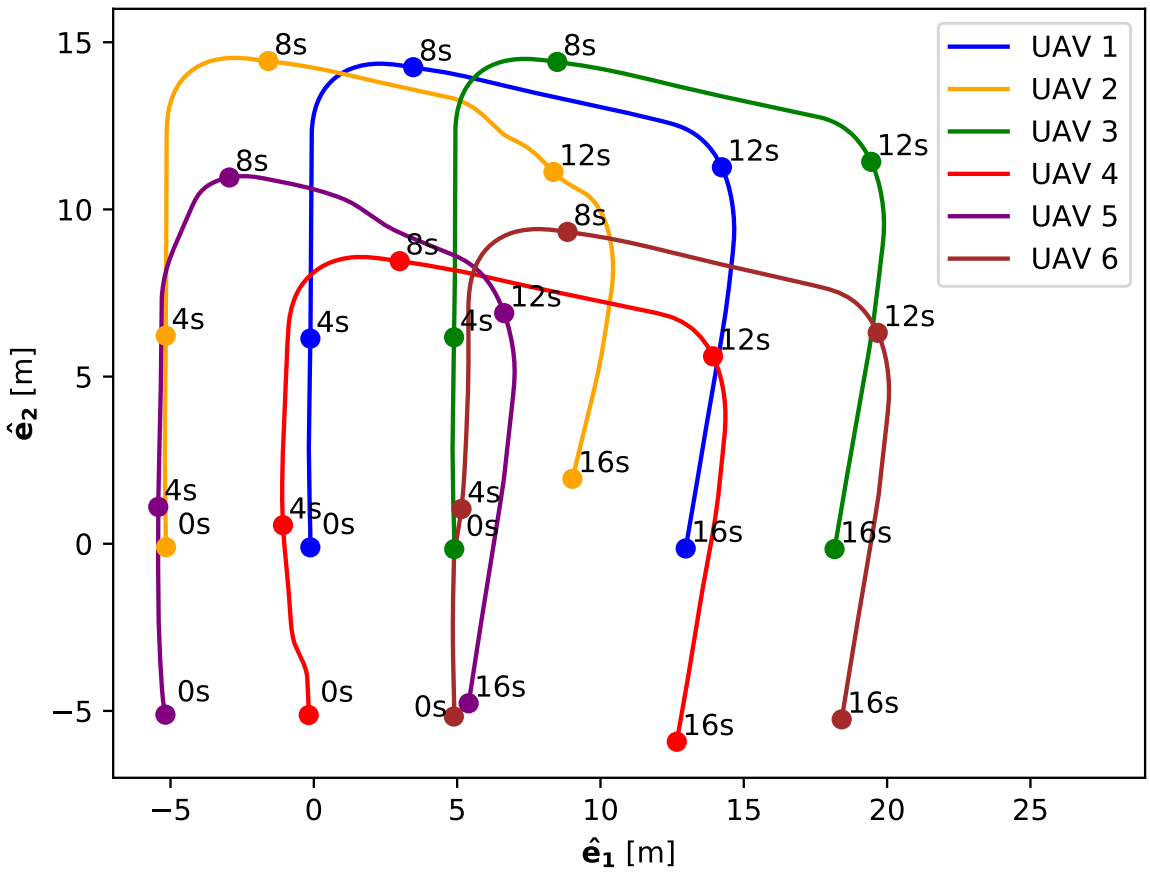


Figure 5.1: Top-down view of the trajectories of the UAVs using the proposed controller. The detection range d_i is set to 5 m for all UAVs.

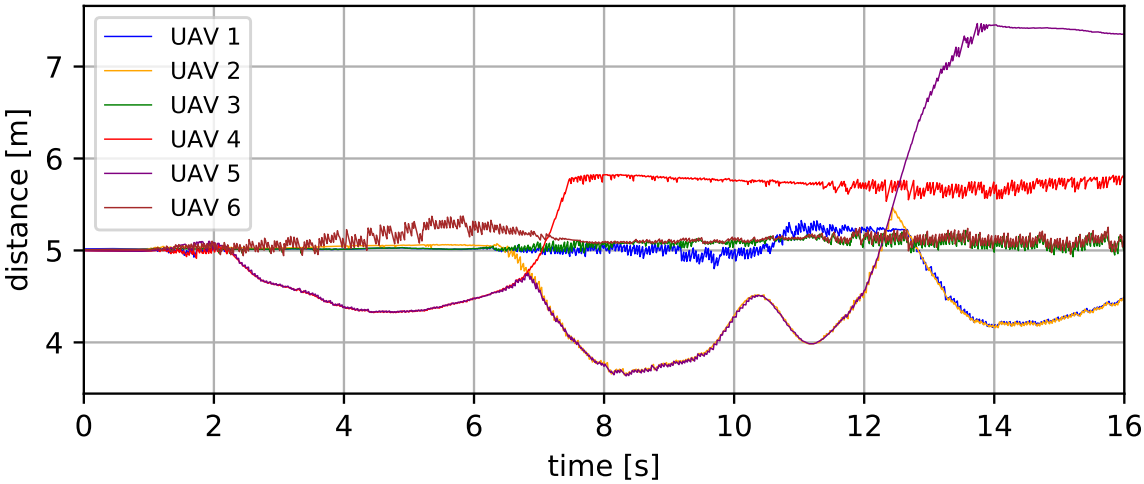


Figure 5.2: Distance to closest UAV. The detection range d_i is set to 5 m for all UAVs.

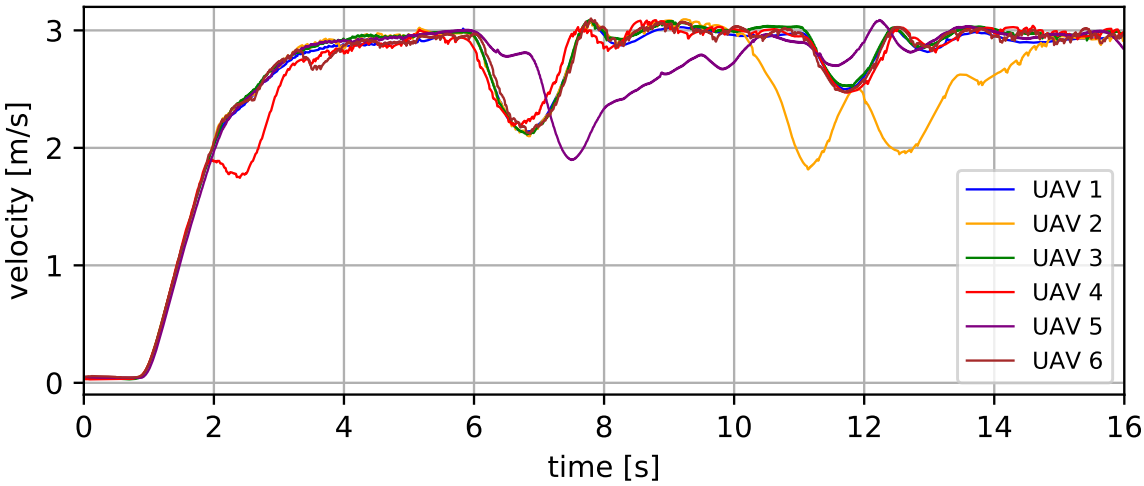


Figure 5.3: Velocities of the UAVs. The detection range d_i is set to 5 m for all UAVs.

5.2 GNSS sharing, longer detection range

In the second test, we increased the detection ranges d_i of all UAVs to 8 meters. We also increased each UAV's safe distance s_i to 5 meters. We left the other settings, including the initial positions of the UAVs, unchanged to evaluate the swarm's behavior under extreme conditions when multiple UAVs have detected other UAV within their detection spherical sectors. The resulting trajectories can be seen in Fig. 5.4.

From Fig. 5.4, it is apparent that the UAVs in the back row immediately detected the UAVs in the front row and started avoiding them. Also, notice that the UAVs 4 and 5 were most likely stuck at their initial positions for a few seconds as they could not fit between UAVs 1, 2, and 6 and UAVs 2 and 4, respectively. UAV 5 probably might have been able to start flying to the left of UAV 2, but UAV 2 further blocked its path as it avoided UAV 1. In Fig. 5.5, we can see that the agents were quick to spread apart and that the distance

to other UAVs decreased only slightly below the initial value, which is a good sign of the swarm's ability to avoid collisions. However, as shown in Fig. 5.5, the minimum distance to other UAVs of the UAVs in the front row greatly exceeded the UAV's detection range d_i of 8 meters. The most likely cause of this is that the UAVs in the front row almost reached their maximum velocity v_{imax} when the UAVs from the back row were still barely moving, as shown in Fig. 5.6, which left the UAVs from the back row far behind. Then, as the UAVs from the front row started moving to the right, they also had to spread among themselves but had to do so at high velocities, which caused the faster UAVs to fly away from the slower ones.

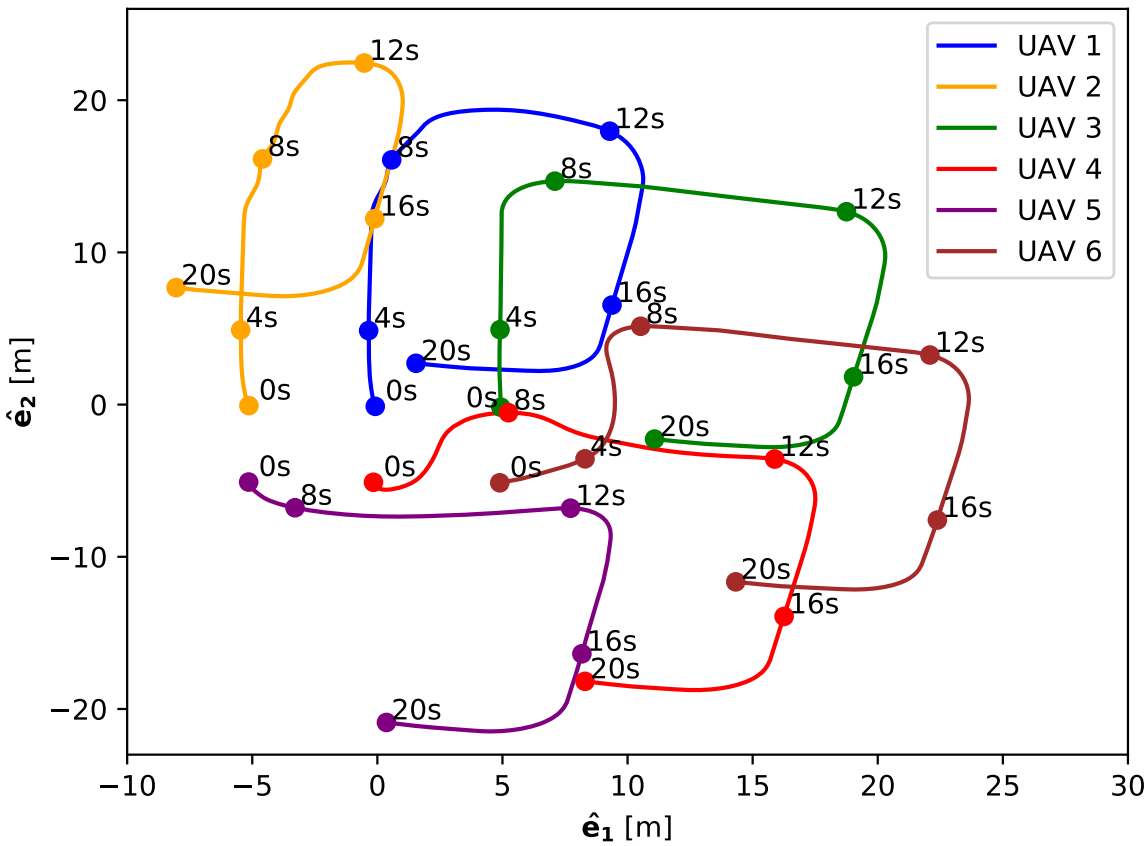


Figure 5.4: Top-down view of the trajectories of the UAVs using the proposed controller. The detection range d_i is set to 8 m for all UAVs.

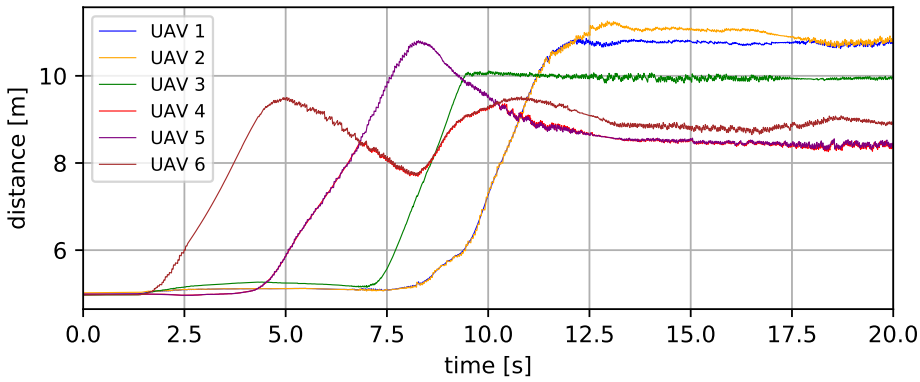


Figure 5.5: Distance to closest UAV. The detection range d_i is set to 8 m for all UAVs.

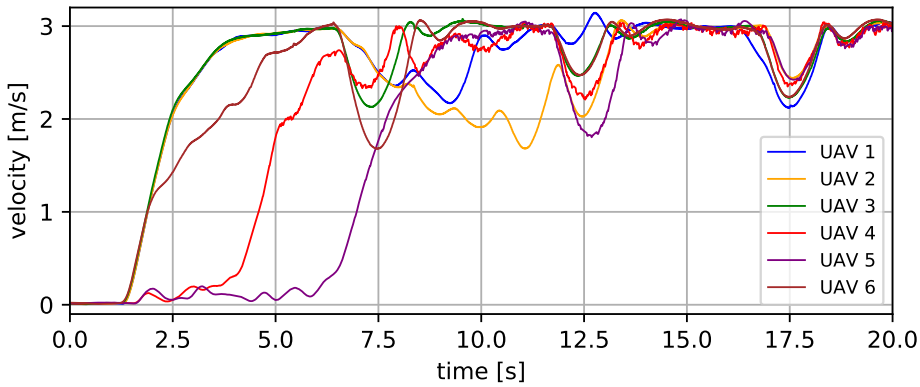


Figure 5.6: Velocities of the UAVs. The detection range d_i is set to 8 m for all UAVs.

5.3 GNSS sharing, common target

In the third test, the settings was identical to the settings in the first test, including the starting positions. The detection range d_i was again set to 5 meters and the safe distance s_i to 2 meters. However, this time, we did not set a separate target for each UAV as would be the case were a human commanding the swarm via gestures. Instead, we set a common target $T = T(i)$ for all the UAVs as in [4]. First, we set the target T's coordinates to $\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} 0 \\ 20 \end{bmatrix}$. After a few seconds, we changed the target T's coordinates to $\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} 20 \\ 20 \end{bmatrix}$, then finally to $\begin{bmatrix} T_x \\ T_y \end{bmatrix} = \begin{bmatrix} 20 \\ 0 \end{bmatrix}$. The resulting trajectories can be seen in Fig. 5.7.

From Fig. 5.7, we can see that the UAVs avoided each other far more frequently than in the previous experiments, which is further confirmed by the noticeably greater fluctuations in velocities, as shown in Fig. 5.9. In Fig. 5.7, at the 8 s mark, the UAVs from the front row were already quite near each other, as also shown in Fig. 5.8. From Fig. 5.8, we can also see that the minimum distance to other UAVs was often lower than the detection range d_i of 5 meters, although the safe distance s_i of 2 meters was never reached. However, what is also apparent from Fig. 5.7 and Fig. 5.8 is that, unlike in the second test, the UAVs that flew farther away from the others eventually returned to the rest of the swarm as they had to reach the same

target T.

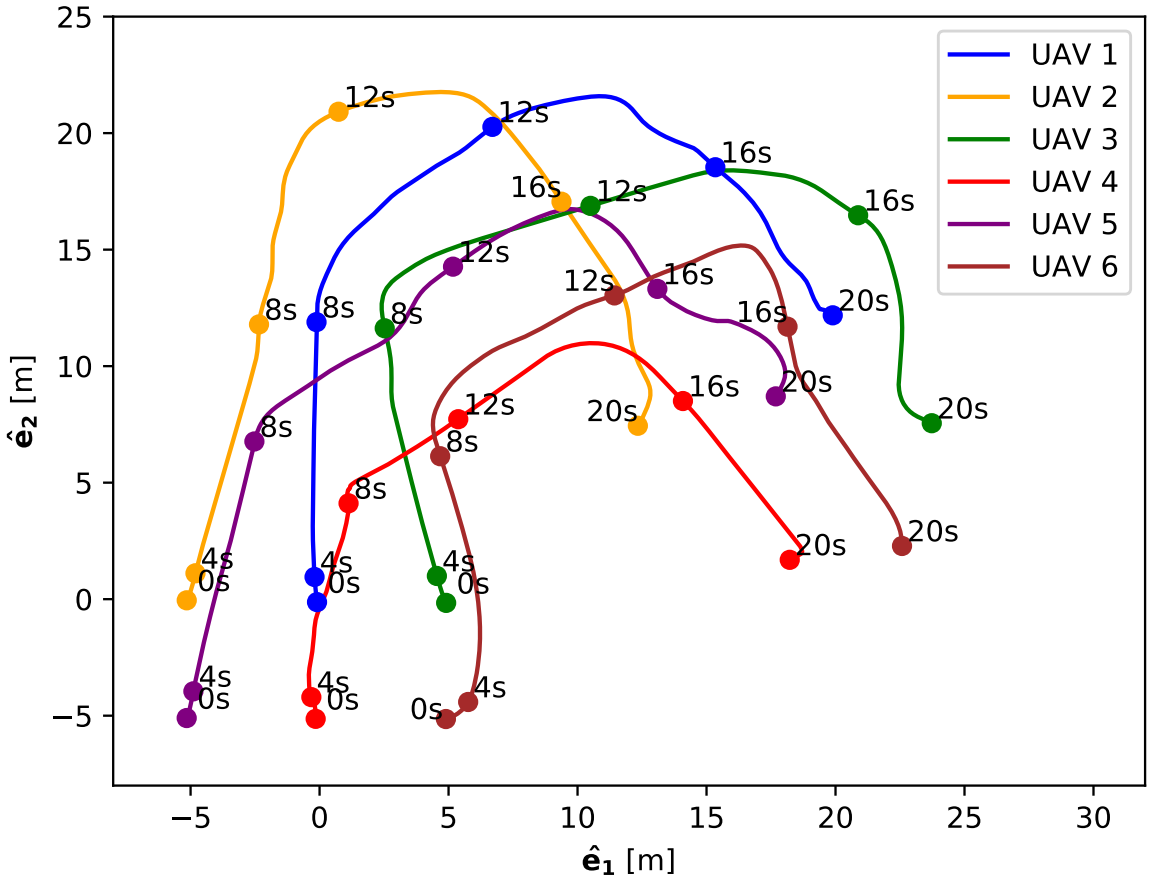


Figure 5.7: Top-down view of the trajectories of the UAVs using the proposed controller. The detection range d_i is set to 5 m for all UAVs and the UAVs share a common target T.

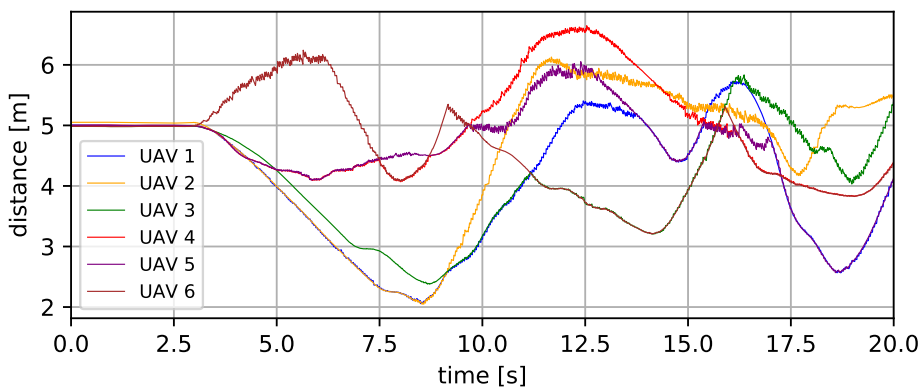


Figure 5.8: Distance to closest UAV. The detection range d_i is set to 5 m for all UAVs and the UAVs share a common target T.

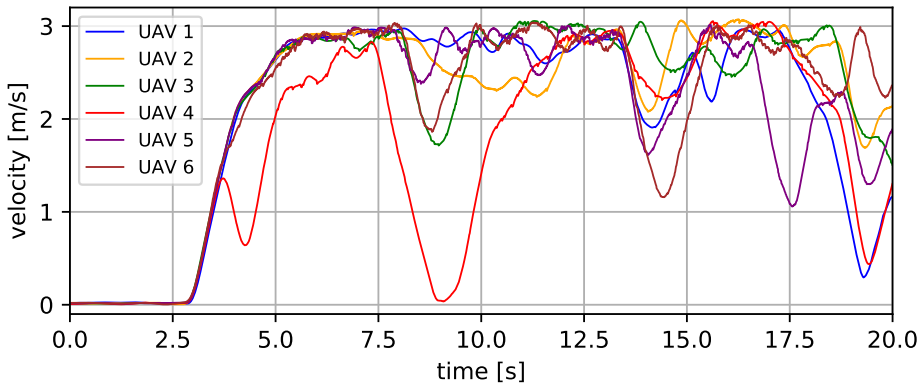


Figure 5.9: Velocities of the UAVs. The detection range d_i is set to 5 m for all UAVs and the UAVs share a common target T.

5.4 UVDAR localization

In the third test, we used the same settings as in the first one, i.e., detection range $d_i = 5\,\mathrm{m}$ and safe distance $s_i = 2\,\mathrm{m}$. However, instead of using implicit communication to share GNSS coordinates among the UAVs for relative localization, the UAVs used GNSS only to measure their own position. The UAVs utilized the novel UVDAR system for relative localization of other UAVs.

The UVDAR system, see [5]–[7], exploits the fact that ultraviolet light frequencies are far less common in nature than visible light and thus are less susceptible to interference. The UAVs are equipped with ultraviolet Light-Emitting Diode (LED) markers and with one or even multiple cameras. In our case, we will be using three cameras. Then, each UAV can use its camera to relatively localize the ultraviolet LED markers of other UAVs in their vicinity. Additionally, the system also allows each UAV to find the identity of the localized UAVs as each UAV can have a distinct blinking pattern of its ultraviolet LED markers.

The trajectories resulting from the simulation are shown in Fig. 5.10. Additionally, the trajectories perceived by individual UAVs are depicted in Fig. 5.12. As we can see from Fig. 5.10, although the settings was identical to the settings in the first test, the trajectories differ significantly. The most likely cause of this is the inaccuracy of the measuring method. For example, in Fig. 5.12d, we can see that 4 seconds into the flight, UAV 4 detects UAV 1 at $\approx \begin{bmatrix} -0.3 \\ 1.7 \end{bmatrix}$, even though, by that time, UAV 1 is already at $\approx \begin{bmatrix} -0.3 \\ 5.7 \end{bmatrix}$, as shown in Fig. 5.12a. Because of that, UAV 4 starts avoiding UAV 1 when, in reality, UAV 1 is outside the detection range of UAV 4. Further, as shown in Fig. 5.11, UAVs 1 and 5 were closer together than the safe distance s_i of 2 meters, if only briefly. In Fig. 5.13, we can see that the velocities of the UAVs were very unstable, which is most likely the result of irregular appearances and disappearances of other UAVs in their detection spherical sector. Perhaps, the system would work better had a lower maximum linear velocity v_{imax} been selected, but that would diminish one of the main features of the proposed controller.

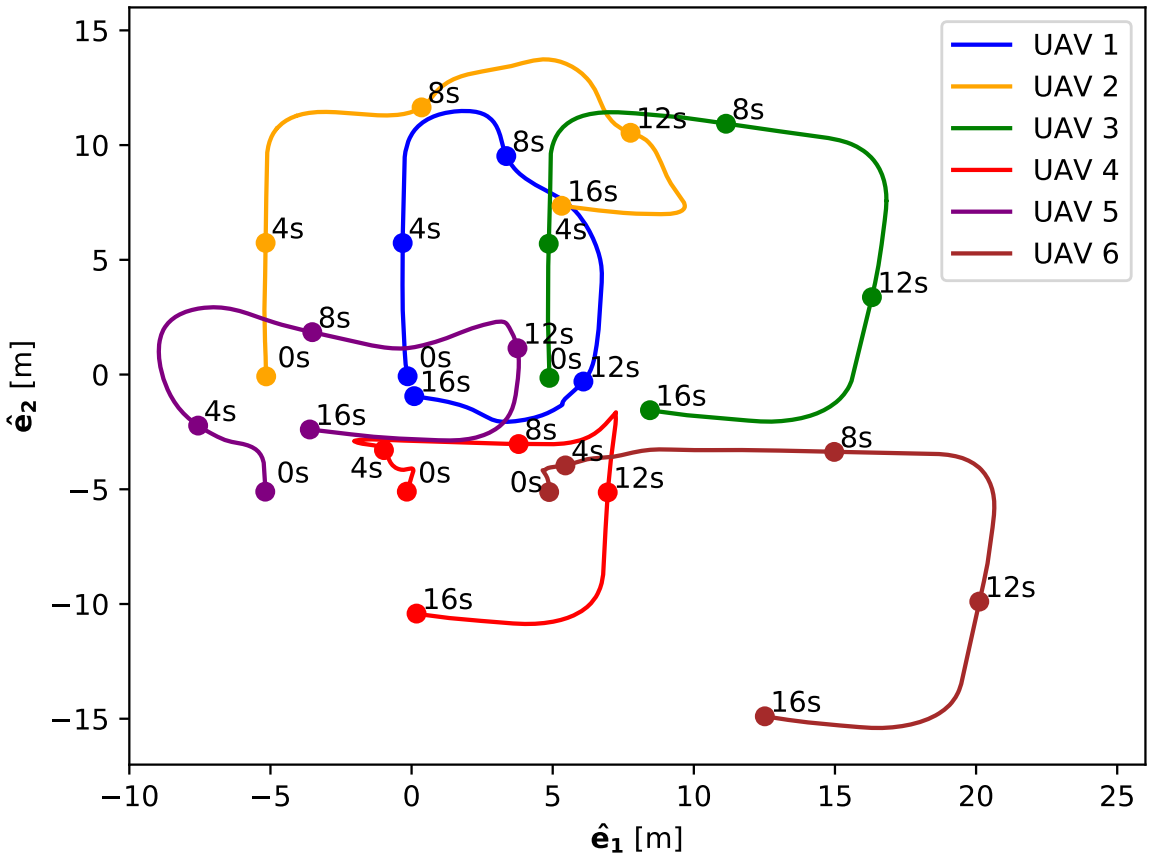


Figure 5.10: Top-down view of the trajectories of the UAVs using the proposed controller. The detection range d_i is set to 5 m for all UAVs and the UAVs use UVDAR for relative localization.

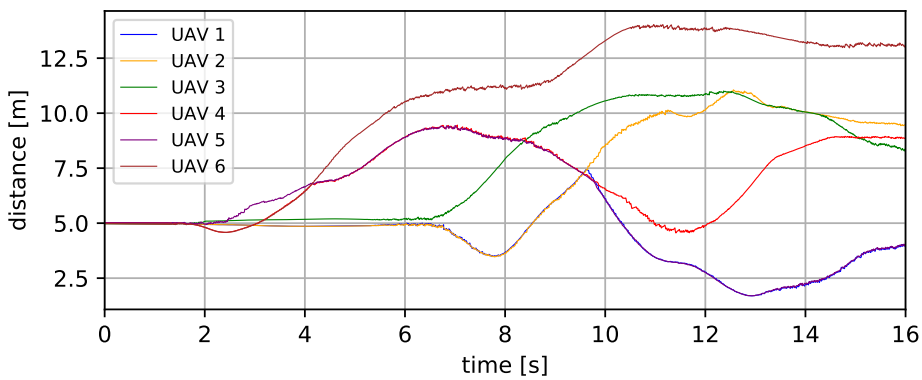
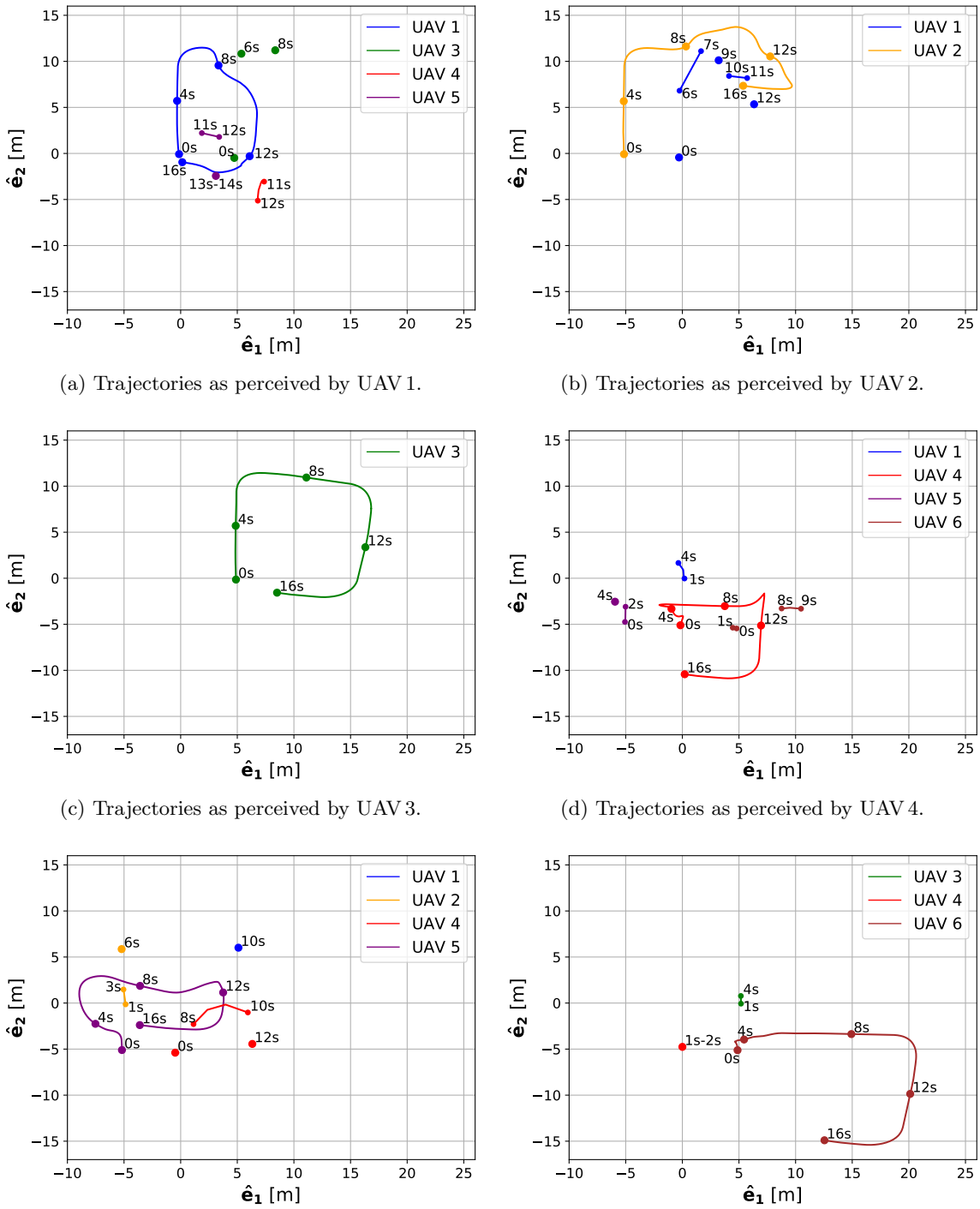


Figure 5.11: Distance to closest UAV. The detection range d_i is set to 5 m for all UAVs and the UAVs use UVDAR for relative localization.



(a) Trajectories as perceived by UAV 1.

(b) Trajectories as perceived by UAV 2.

(c) Trajectories as perceived by UAV 3.

(d) Trajectories as perceived by UAV 4.

(e) Trajectories as perceived by UAV 5.

(f) Trajectories as perceived by UAV 6.

Figure 5.12: Trajectories of the UAVs as perceived by each UAV. Each UAV gets its own position using GNSS, but the relative position of others is obtained via UVDAR and then transformed to GNSS coordinates. Only UAVs within a detection spherical sector are included.

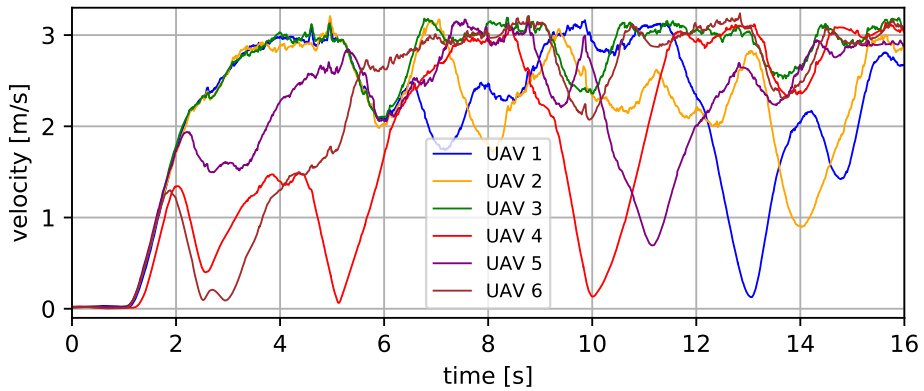


Figure 5.13: Velocities of the UAVs. The detection range d_i is set to 5 m for all UAVs and the UAVs use UVDAR for relative localization.

5.5 Flocking approaches comparison

Apart from testing the proposed controller, we also performed a few tests using the flocking controller from [1], which is based on the Boids controller proposed in [22], so that we can make a comparison between the two approaches. The approach from [1] works by combining three virtual forces (also called vectors). These virtual forces are the proximal force, which keeps the swarm together, the navigation force, which moves agents informed about a goal toward the goal, and the collision force responsible for collision avoidance. For simplicity reasons, we did not use the navigation force. Instead, we used an approach where UAV 1 was unaware of the swarm. UAV 1 was sent toward the goal and the other UAVs, which used the controller from [1], followed. During the test, UAV 1 was initially left standing for a few seconds, then sent toward the position $\begin{bmatrix} 0 \\ 20 \end{bmatrix}$, where UAV 1 stayed for a few more seconds. Then, UAV 1 was sent to the position $\begin{bmatrix} 20 \\ 20 \end{bmatrix}$. After a delay, the UAV was sent to the final position $\begin{bmatrix} 20 \\ 0 \end{bmatrix}$, where the UAV stayed until the end. We performed the test twice, each time using a different proximity coefficient k_{prox}, which adjusts how much the proximity force pulls the UAVs together.

5.5.1 Lower proximity coefficient

For the first test utilizing the controller from [1], we set the proximity coefficient $k_{prox} = 1.5$. The resulting trajectories are shown in Fig. 5.14. Compared to the proposed controller, the trajectories are most similar to the case with a common target T in Sec. 5.3. The minimum distance to other UAVs in Fig. 5.15 is also most comparable to the case with a common target T. The minimum distance is relatively low, and the UAVs which fly farther away from the others eventually return. However, the minimum distance to other UAVs is much more stable when using the controller from [1] than when using the proposed controller. This is also true for the velocities, shown in Fig. 5.16, where there are noticeable disturbances at the beginning before the UAVs manage to organize themselves and then slight increases when the leading UAV 1 flies farther away from the others, but, other than that, the velocities are pretty stable.

From Fig. 5.16, an obvious drawback is apparent: the velocities are way lower than when using the proposed controller. When using the controller from [1], velocity can be adjusted in two ways. Firstly, by increasing the number of informed agents. When more agents are aware of the goal and thus are moving toward it, the proximity force affecting the rest of the agents is stronger, so they move faster. Secondly, velocity can be increased by making the proximity force stronger in itself, i.e., by increasing the proximity coefficient k_{prox}. We will look into the second option in Sec. 5.5.2.

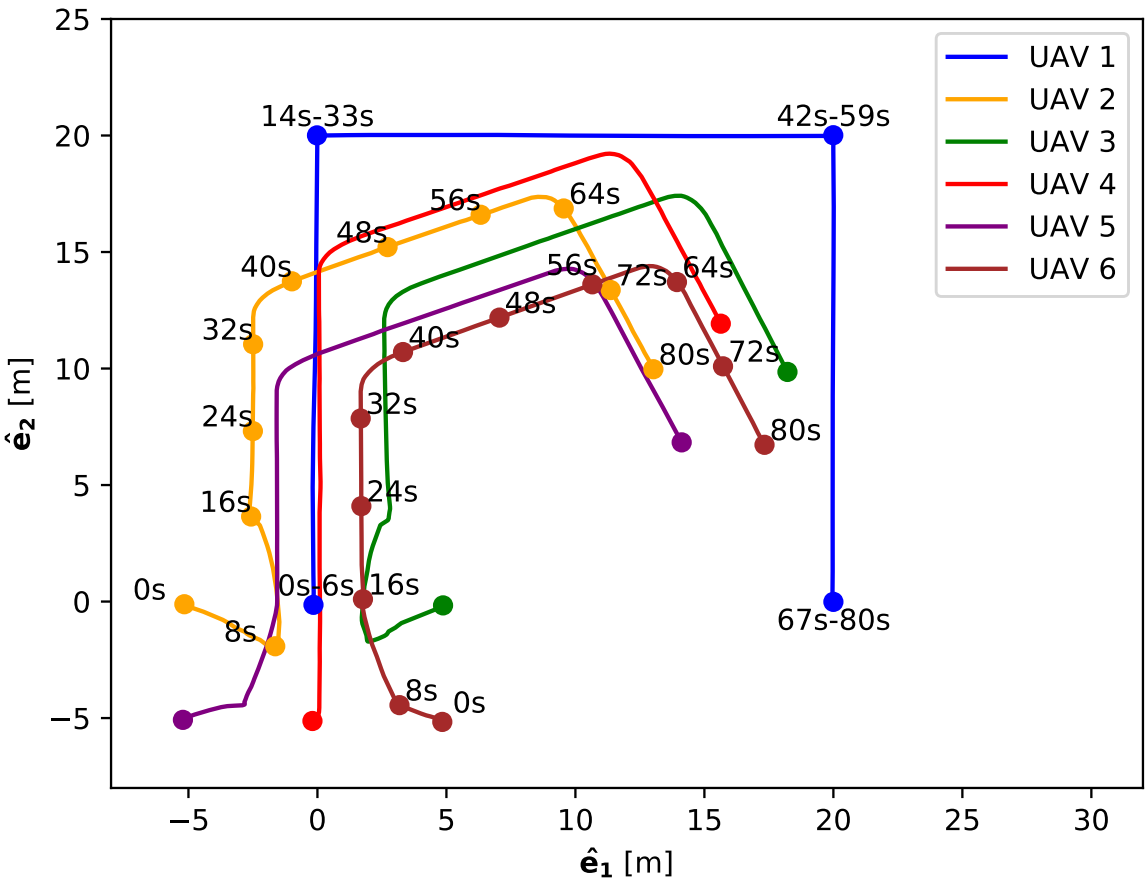


Figure 5.14: Top-down view of the trajectories of the UAVs using the controller from [1] with the proximity coefficient set to 1.5.

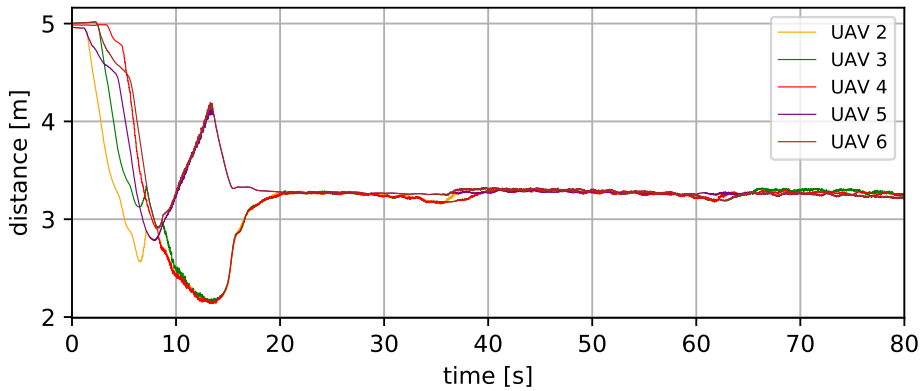


Figure 5.15: Distance to closest UAV when using the controller from [1] with the proximity coefficient set to 1.5. The leading UAV 1 is excluded from the graph.

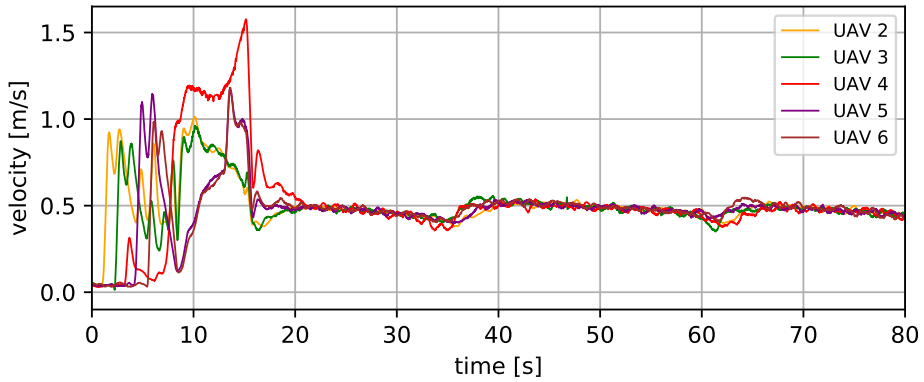


Figure 5.16: Velocities of the UAVs using the controller from [1] with the proximity coefficient set to 1.5. The leading UAV 1 is excluded from the graph.

5.5.2 Higher proximity coefficient

In the second test using the controller from [1], we increased the proximity coefficient k_{prox} to 3.0. The resulting trajectories can be seen in Fig. 5.17. In Fig. 5.17, it is apparent that the UAVs were not as successful at moving steadily as in the previous test using the lower proximity coefficient k_{prox}, which is especially visible at the start of the flight, where they had more trouble organizing themselves. Although the mean velocity almost doubled from $\approx 0.54\,\mathrm{m\,s^{-1}}$ to $\approx 0.97\,\mathrm{m\,s^{-1}}$, we can see that neither the velocity, shown in Fig. 5.19, nor the minimum distance to another UAV, in Fig. 5.18, were as stable as in the previous test in Sec. 5.5.1. Additionally, the velocities of the UAVs when using the controller from [1] are still far lower compared to the velocities of the UAVs using the controller proposed in this thesis.

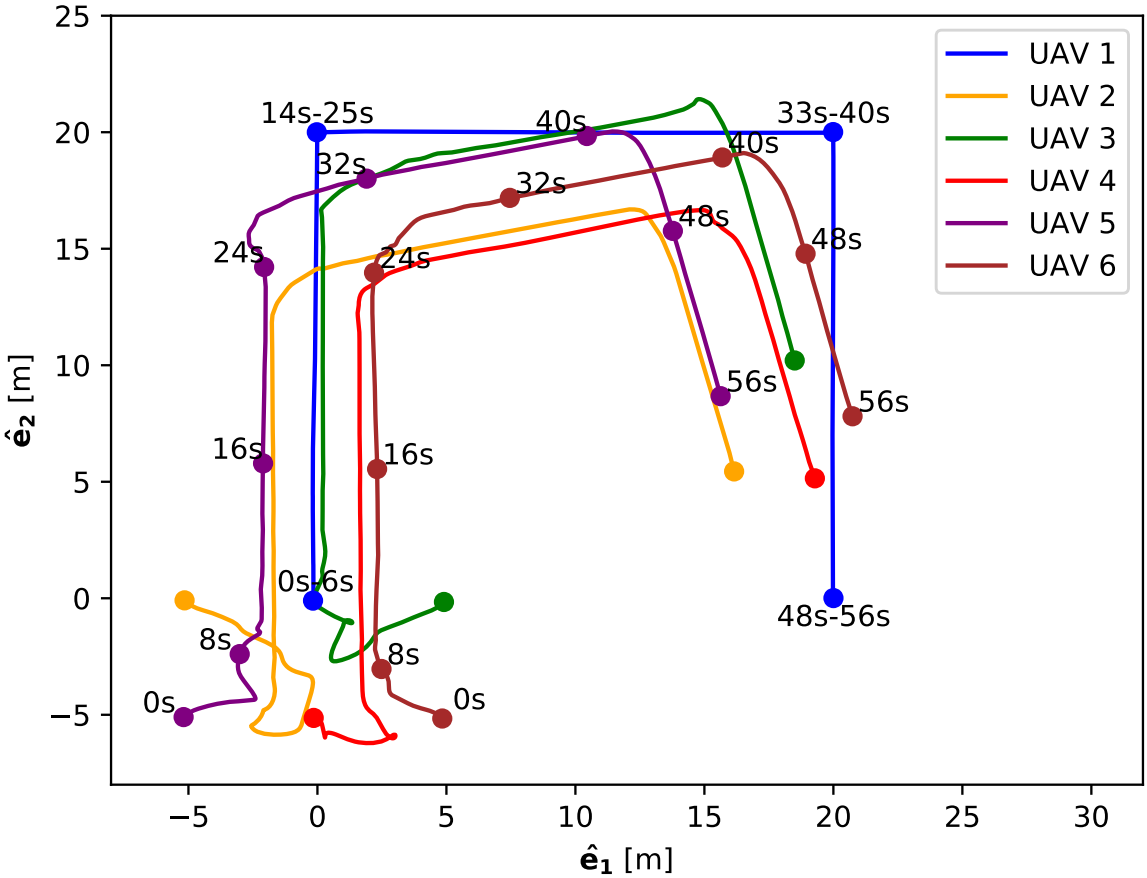


Figure 5.17: Top-down view of the trajectories of the UAVs using the controller from [1] with the proximity coefficient set to 3.0.

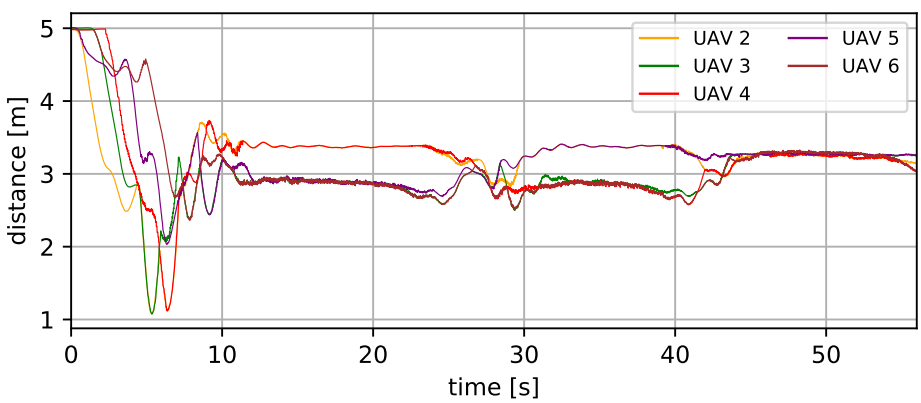


Figure 5.18: Distance to closest UAV when using the controller from [1] with the proximity coefficient set to 3.0. The leading UAV 1 is excluded from the graph.

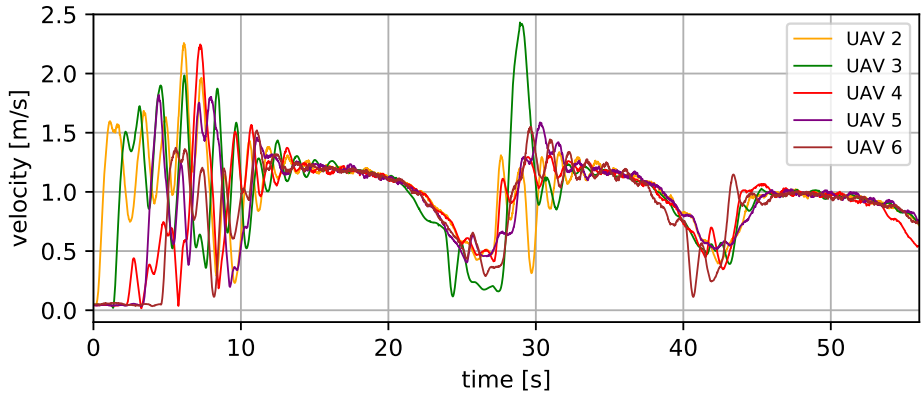


Figure 5.19: Velocities of the UAVs using the controller from [1] with the proximity coefficient set to 3.0. The leading UAV 1 is excluded from the graph.

5.5.3 Comparison

The simulations show that the controller from [1] has superior properties in terms of cohesion, and in operating the swarm as a single unit rather than multiple individuals. This results in more stable velocities (Fig. 5.16) and minimum distances to other UAVs (Fig. 5.15) when compared to the proposed controller (Fig. 5.3 and Fig. 5.2). Such behavior can be necessary for some applications, such as search and rescue operations when a single cohesive swarm can search a wide area more efficiently. Additionally, the agents can then cooperate to more reliably identify the target, as proposed in [19].

However, the strong cohesion of the controller from [1] is also the cause of its disadvantage: slow movement. The velocity of a UAV in a swarm controlled by the controller from [1] directly depends on the number of agents informed about the goal. The velocity can also be tuned by adjusting the influence of the proximity force on the UAVs, though this may lead to instability and oscillations in the swarm. Further, a single slow member of the swarm will also slow down the rest of the swarm, even if all the agents are informed about the goal. Additionally, the slowdown of the swarm caused by the slow member will be even more significant when having a stronger proximity force. Another undesirable trait of the controller from [1] is its oscillations. When the UAVs are farther apart, the proximity force pulls them together. Upon reaching a certain threshold in the distances between the UAVs, the collision force becomes dominant and spreads the UAVs apart, which again causes the proximity force to pull the UAVs together, repeating indefinitely. Thus, the UAVs oscillate even when not moving toward a goal.

The flocking controller proposed in this thesis does not suffer from the defects mentioned above. Because there is no explicit cohesion in the system, the UAVs will fly toward their target as fast as possible, no matter how many other UAVs are aware of the target or how fast the other UAVs move. This property can be advantageous in scenarios where high velocity is required, such as chasing a foreign UAV which entered a restricted area. Another advantage of the proposed controller is its scalability. Because each UAV needs to consider only the closest other UAV in a spherical sector in front of itself, the number of UAVs in the swarm can easily be raised without having a major impact on a UAV's required computing power.

5.6 Left command

In this section and the following sections, we will present the results from tests run to verify the behavior of human-swarm interaction. Whenever a human is present in the results among the UAVs, its position is taken as perceived by the UAVs. Table 5.2 shows the parameter settings used for the following tests. The initial positions of the UAVs will be in a similar grid as in the previous tests, except that the row and column spacing might differ. Additionally, UAV 3 was removed; therefore, there will only be 5 UAVs in each test.

Parameter	Symbol	Value
Maximum linear velocity	v_{imax}	differs per test
Safe distance	s_i	2 m
Detection range	d_i	5 m
Detection angle	φ_i	$1.27 \approx \frac{\pi}{2} - 0.3$
Target radius	$T_r(i)$	8.0 m
Free subsystem virtual heading rate factor	k_f	3.0
Engaged subsystem virtual heading rate factor	k_e	6.2
Engaged subsystem deceleration factor	λ_i	$3.2\,\mathrm{m\,s^{-2}}$
Real heading rate factor	k_h	3.0
Gesture consensus timer	t_{iC}	1.0 s
Distance to fly	d_f	20.0 m

Table 5.2: Parameter settings used for human-swarm interaction simulation tests.

In the first human-swarm interaction test, the goal was to verify that the UAVs correctly identified the human's gesture and executed the appropriate command simultaneously. The UAVs were spaced 5 meters apart, and their maximum linear velocity v_{imax} was set to $3\,\mathrm{m\,s^{-1}}$. A few seconds after the start of the test, a human was spawned at $\begin{bmatrix}0\\8\end{bmatrix}$ with the yaw angle set to $-\frac{\pi}{2}$. A few seconds later, we set the human's gesture to Left.

As shown in Fig. 5.23 and Fig. 5.20, the UAVs detected the human, turned their real heading accordingly, then executed the correct command. Fig. 5.20 further reveals that the UAVs properly adjusted their real headings to keep sight of the human throughout the flight. Fig. 5.21 and Fig. 5.22 show that about 7 seconds after the start, UAV 6 slightly moved, probably due to some instability in the lower-level controls. However, the slight movement shifted UAV 6 somewhat closer to UAV 4, which later caused UAV 6 to circle UAV 4, as shown in Fig. 5.23. Apart from UAV 6, the UAVs managed to keep almost constant distances from one another, as shown in Fig. 5.21. The sudden changes in the human's distance, seen in Fig. 5.21, result from the UAVs combining their observations of the human as described in (4.3) and (4.4). However, the velocities of the UAVs, displayed in Fig. 5.22, reveal an undesired behavior: the UAVs' initial acceleration is constant, which is in contrast with the velocity calculation introduced in (3.8). Further analysis revealed the underlying problem to be the UAVs staying too long in the *free subsystem*, which increases the value of t_f^i used in (3.8). When the UAVs start moving, their calculated linear velocity v_i has already approached the maximum linear velocity v_{imax}.

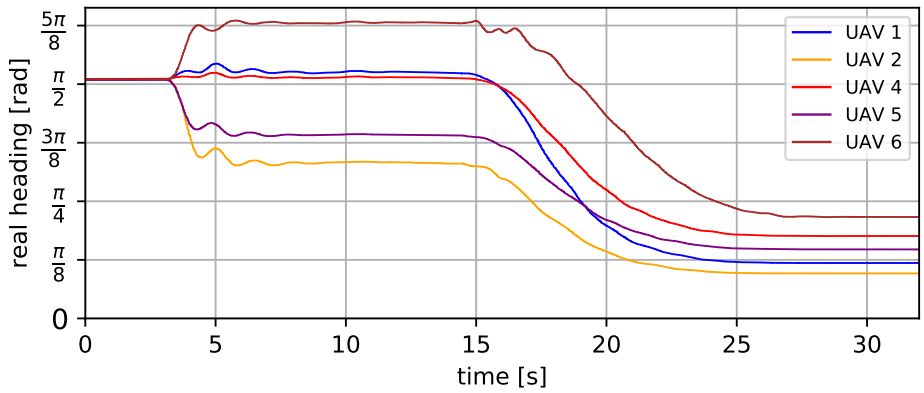


Figure 5.20: Real headings of the UAVs when reacting to the gesture Left.

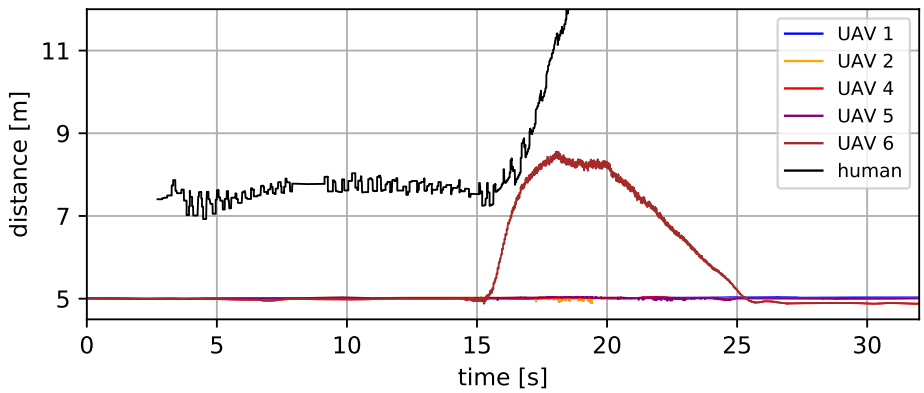


Figure 5.21: Distance to closest UAV or human when reacting to the gesture Left.

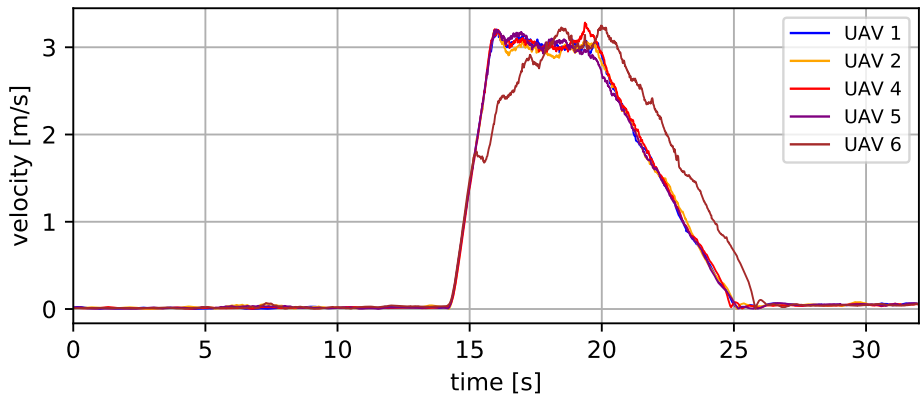


Figure 5.22: Velocities of the UAVs when reacting to the gesture Left.

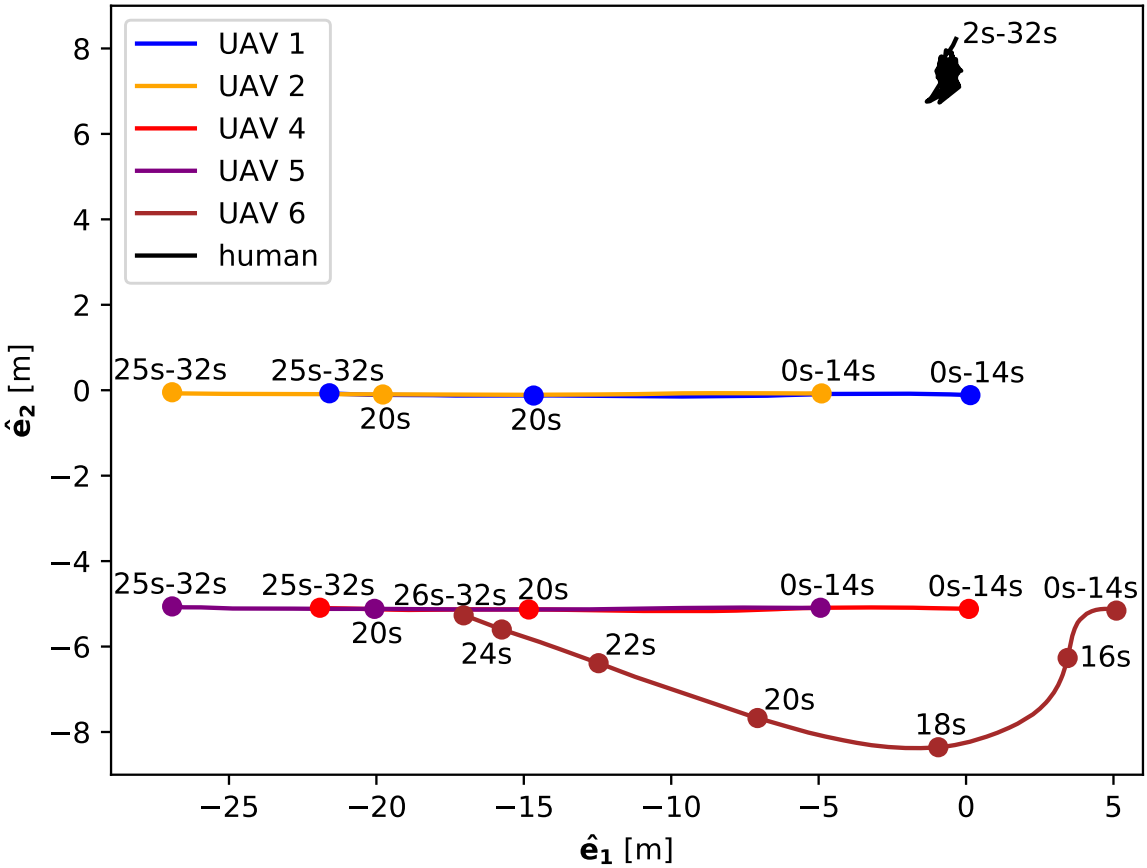


Figure 5.23: Top-down view of the trajectories of the UAVs and the human when reacting to the gesture Left.

5.7 Avoiding the human

Sec. 4.6 proposes an approach to avoiding the human operator based on placing a virtual UAV at the human's position. The UAVs in the swarm should then avoid the human as they would avoid a UAV.

To verify the described behavior, we spaced the UAVs 6 meters apart and set their maximum linear velocity v_{imax} to $3\,\mathrm{m\,s^{-1}}$. A few seconds after the start, we spawned a human at $\begin{bmatrix}0\\8\end{bmatrix}$ with the yaw angle set to $-\frac{\pi}{2}$. A few seconds later, we set the human's gesture to Forward. A video from this test is available online at `http://mrs.felk.cvut.cz/prazak-2022-bp`.

As depicted in Fig. 5.27, the UAVs reacted to the gesture correctly, and UAVs 1 and 4 actively avoided the human. Fig. 5.25 shows that the UAVs kept their distance from the human greater than the safe distance $s_i = 2\,\mathrm{m}$. However, as the UAVs 1 and 4 avoided the human, they neared the other UAVs. About 25 seconds into the flight, UAV 6 detected UAV 1, and UAV 4 detected UAV 5. As shown in Fig. 5.26, UAV 4 had to decrease its linear velocity v_4 almost to zero as it had to fit between the human and UAV 5. Similar to the previous test, the UAVs successfully observed the human's position throughout the flight, as displayed in Fig. 5.24.

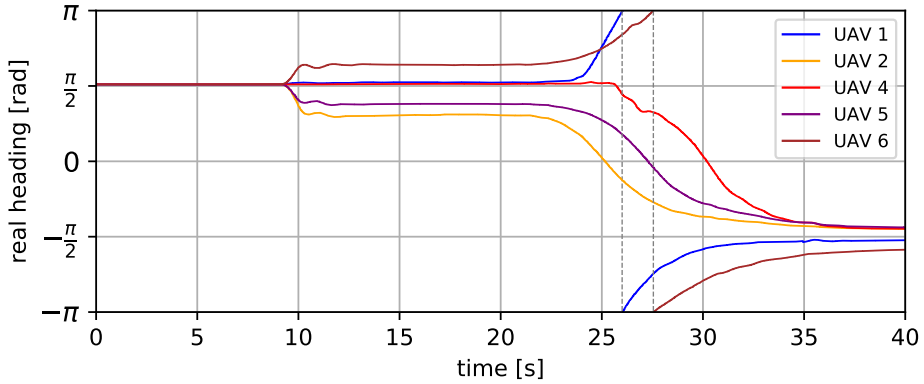


Figure 5.24: Real headings of the UAVs. The UAVs are avoiding the human as they react to the gesture Forward.

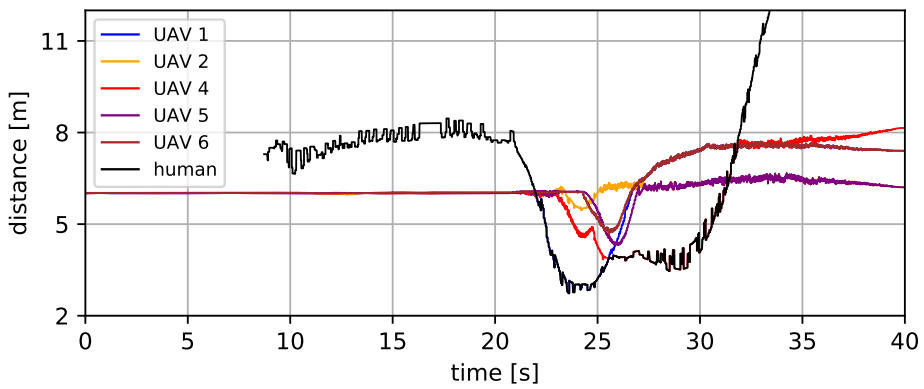


Figure 5.25: Distance to closest UAV or human. The UAVs are avoiding the human as they react to the gesture Forward.

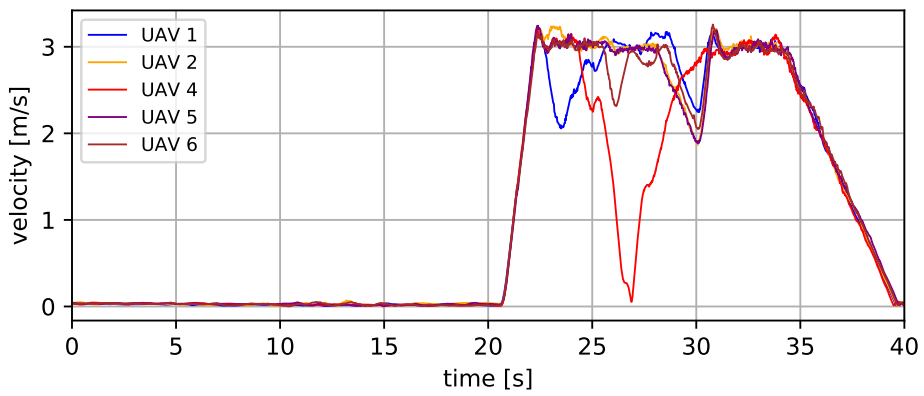


Figure 5.26: Velocities of the UAVs. The UAVs are avoiding the human as they react to the gesture Forward.

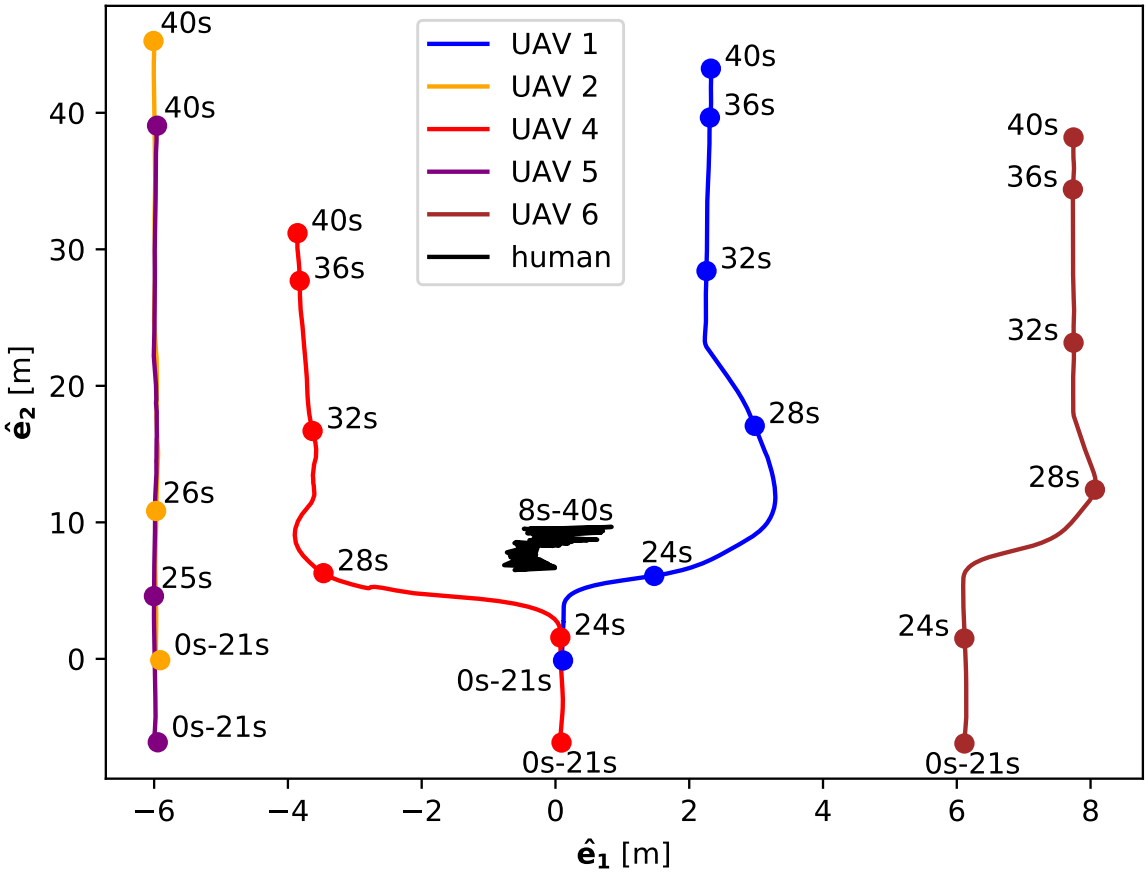


Figure 5.27: Top-down view of the trajectories of the UAVs and the human. The UAVs are avoiding the human as they react to the gesture Forward. A video from the test is available at `http://mrs.felk.cvut.cz/prazak-2022-bp`.

5.8 Changing gesture

The last simulation test verifies that the swarm correctly handles a situation when the human's gesture changes during the flight. We spaced the UAVs 6 meters apart and set their maximum linear velocity v_{imax} to $2\,\mathrm{m\,s^{-1}}$. A few seconds after the start of the test, we spawned a human at $\begin{bmatrix}-13\\5\end{bmatrix}$ with the yaw angle set to $-\frac{\pi}{4}$. After a few seconds, we set the human's gesture to Forward. When the swarm started moving, we changed the human's gesture to Left. A video from this test is available online at `http://mrs.felk.cvut.cz/prazak-2022-bp`.

The trajectories depicted in Fig. 5.31 show that the UAVs recognized both gestures and executed the proper commands. The UAVs correctly calculated the new target coordinates according to their current position. The distances in Fig. 5.29 confirm that the UAVs managed to avoid both the human and one another. The UAVs did not even lose much velocity when reacting to the gesture change and when avoiding the human, as shown in Fig. 5.30. Fig. 5.28 shows that the UAVs managed to keep observing the human midflight as in the previous tests.

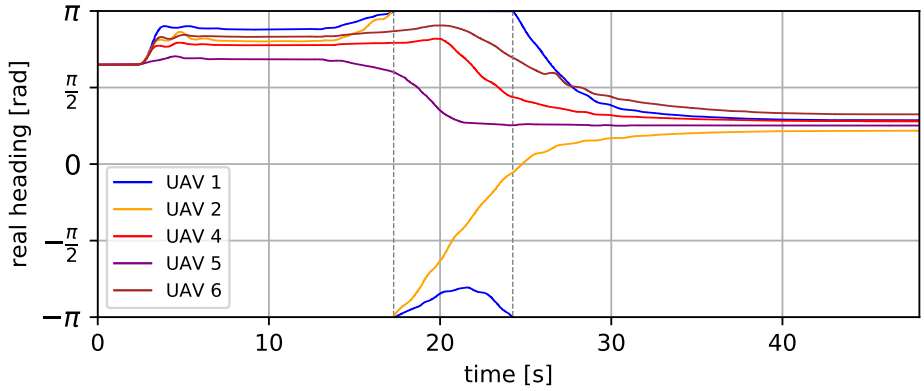


Figure 5.28: Real headings of the UAVs. The UAVs are reacting to changing gestures.

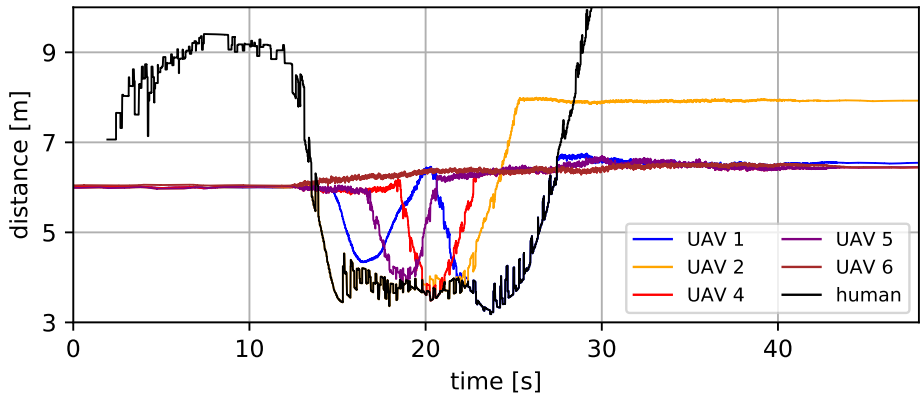


Figure 5.29: Distance to closest UAV or human. The UAVs are reacting to changing gestures.

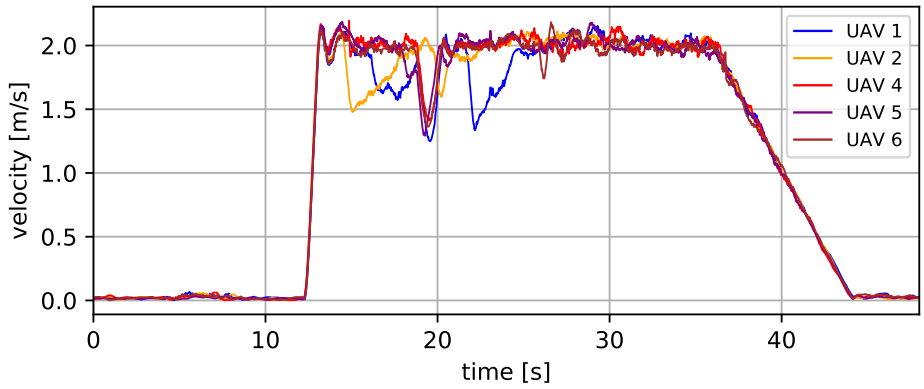


Figure 5.30: Velocities of the UAVs. The UAVs are reacting to changing gestures.

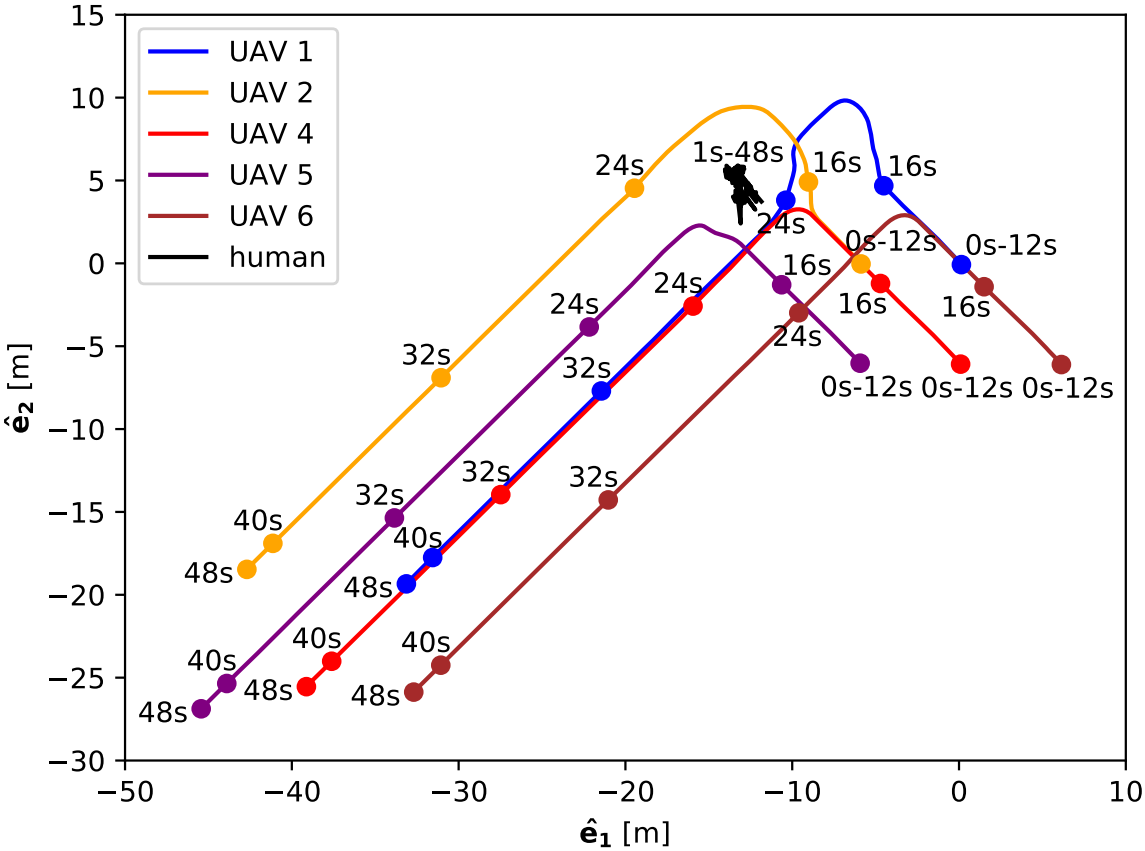


Figure 5.31: Top-down view of the trajectories of the UAVs and the human. The UAVs are reacting to changing gestures. A video from the test is available at `http://mrs.felk.cvut.cz/prazak-2022-bp`.

Chapter 6

Hardware experiments

In this chapter, we will go through the results gathered from real-world experiments utilizing the proposed flocking controller and human-swarm interaction. The experiments were conducted using the DJI f450 UAVs of the MRS[1]. Each UAV in the swarm was equipped with a Global Positioning System (GPS) receiver for localization, WiFi receiver and transmitter for inter-agent communication, and an Intel RealSense[2] camera for pose estimation and gesture recognition. One of the UAVs used for the experiments, namely UAV 47, is pictured in Fig. 6.1.

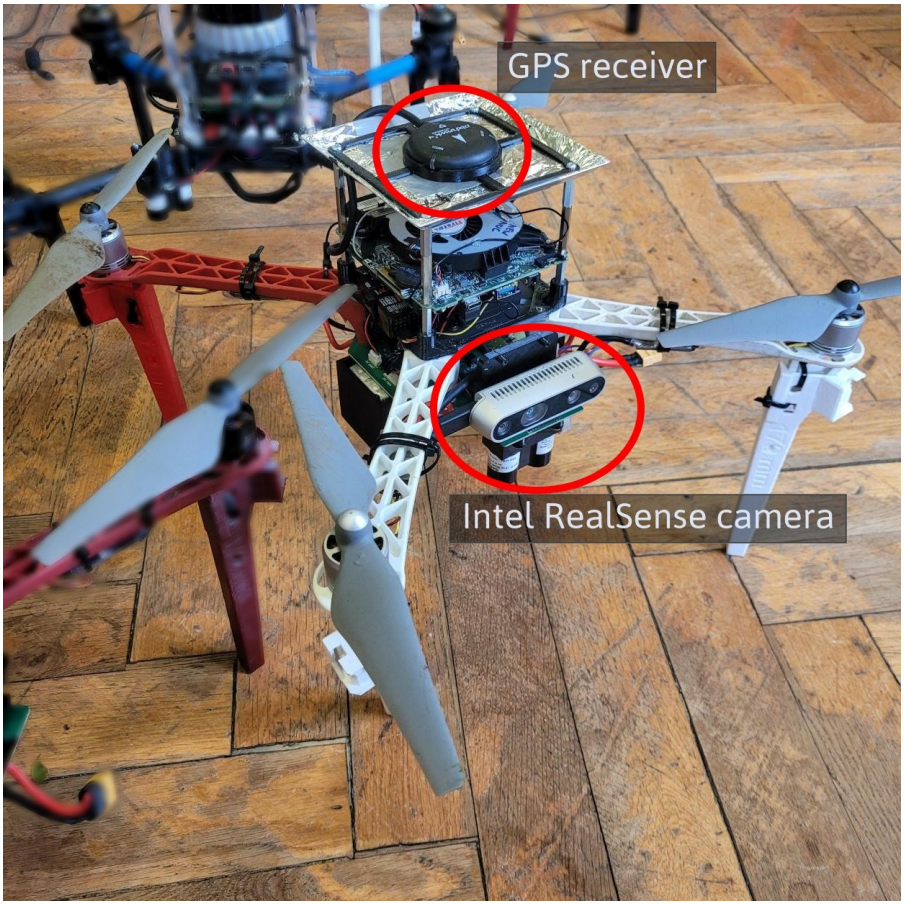


Figure 6.1: Detail of UAV 47, which was used for hardware experiments. GPS receiver and Intel RealSense camera are highlighted in the picture.

[1] https://github.com/ctu-mrs/mrs_uav_system#unmanned-aerial-vehicles
[2] https://www.intelrealsense.com/

The parameter settings used for the experiments is shown in Table 6.1.

Parameter	Symbol	Value
Maximum linear velocity	v_{imax}	differs per test
Safe distance	s_i	5 m
Detection range	d_i	8 m
Detection angle	φ_i	$1.27 \approx \frac{\pi}{2} - 0.3$
Target radius	$T_r(i)$	8.0 m
Free subsystem virtual heading rate factor	k_f	3.0
Engaged subsystem virtual heading rate factor	k_e	6.2
Engaged subsystem deceleration factor	λ_i	$3.2\,\mathrm{m\,s^{-2}}$
Real heading rate factor	k_h	3.0
Gesture consensus timer	t_{iC}	1.0 s
Distance to fly	d_f	10.0 m

Table 6.1: Parameter settings used for real-world experiments.

6.1 Swarm flight

In the first experiment, the swarm consisted of three UAVs. We set the maximum linear velocity v_{imax} to $1\,\mathrm{m\,s^{-1}}$ and let the swarm fly similarly as in the first simulation test in Sec. 5.1. We set each UAV's target $T(i)$ relative to the UAV's location $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$ according to the formula $\begin{bmatrix} T_x(i) \\ T_y(i) \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} \delta_x(k) \\ \delta_y(k) \end{bmatrix}$, waited a few seconds, then repeated the process with k incremented by 1. Values of $\begin{bmatrix} \delta_x(k) \\ \delta_y(k) \end{bmatrix}$ are in Table 6.2. The resulting trajectories are shown in Fig. 6.2.

Target offset	k - target number			
	1	2	3	4
$\delta_x(k)$	7	20	0	-30
$\delta_y(k)$	20	-5	-20	-5

Table 6.2: Target offsets used for first hardware experiment.

From Fig. 6.2, we can see that the UAVs successfully followed their targets and maintained formation. About 72 seconds into the flight, UAV 61 slightly adjusted its path when it detected UAV 60, also shown in Fig. 6.3. Further, about 197 seconds into the flight, UAV 60 briefly detected UAV 61. Other than that, the UAVs stayed outside one another's detection spherical sectors. Interestingly, the maximum linear velocity v_{imax}, shown in Fig. 6.4, was often exceeded. This was most likely caused by environmental disturbances, such as wind. The noticeable oscillations of the UAVs' velocities suggest that the controller unsuccessfully attempted to keep the UAVs at their maximum linear velocity v_{imax}, which was not seen during the simulation tests, e.g., in Fig. 5.3. The linear deceleration, introduced in (3.9), is also apparent in Fig. 6.4, apart from the flight toward the last target, which was ended prematurely by a human operator.

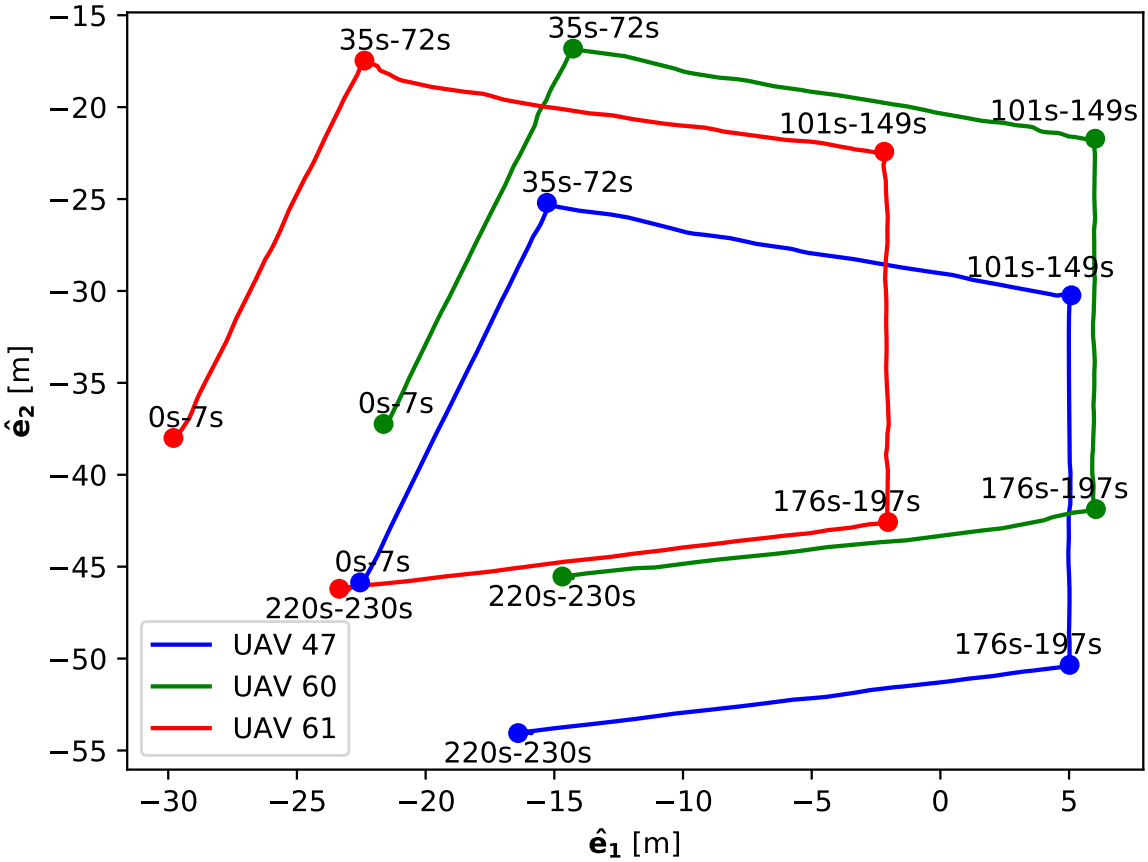


Figure 6.2: Top-down view of the trajectories of the UAVs using the proposed controller during the swarm flight real-world experiment.

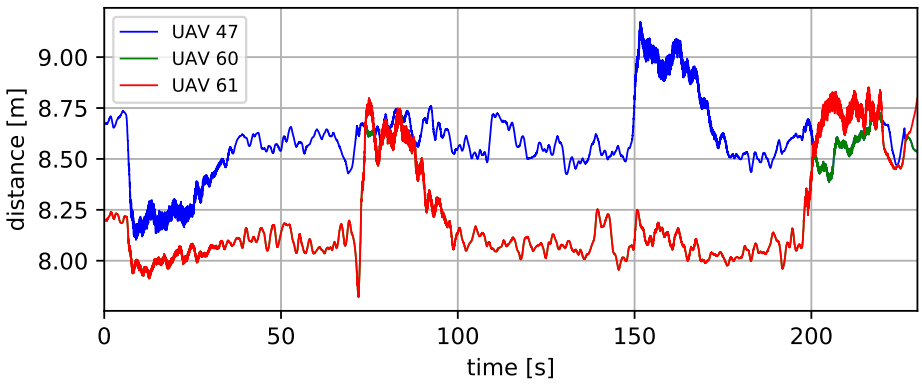


Figure 6.3: Distance to closest UAV during the swarm flight real-world experiment.

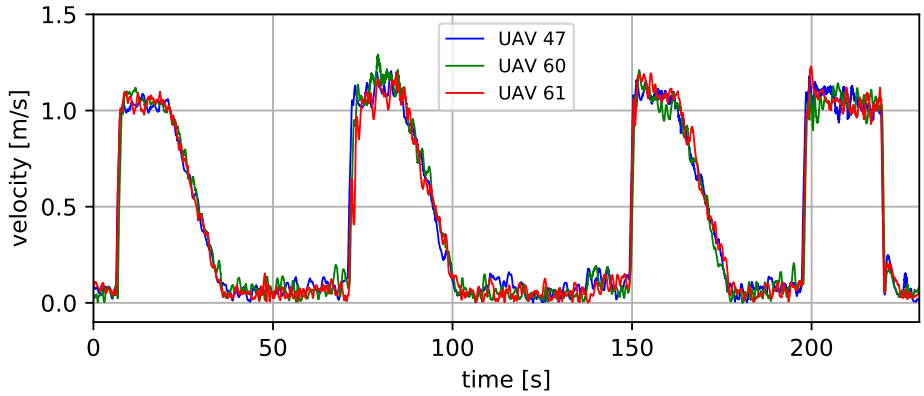


Figure 6.4: Velocities of the UAVs during the swarm flight real-world experiment.

6.2 UAV avoidance at lower velocity

In the second experiment, we wanted to see how a UAV circles a stationary UAV. The swarm consisted of two UAVs - UAV 47 and UAV 60. Both had the proposed controller enabled; however, only UAV 47 was aware of a target $T(47)$. Additionally, UAV 60 blocked the direct path from UAV 47 to the target $T(47)$. A few seconds after UAV 47 reached target $T(47)$, the target was moved a few meters away from the original starting position of UAV 47. Therefore, UAV 47 performed the maneuver twice, once in each direction. The maximum linear velocity v_{imax} was set to $1\,\mathrm{m\,s^{-1}}$. The resulting trajectory is shown in Fig. 6.6.

The trajectory presented in Fig. 6.6 is smooth and shows that UAV 47 avoided UAV 60 without difficulties. When we look at the distance between UAV 47 and UAV 60 in Fig. 6.5, it is evident that the UAVs were much closer than the detection range $d_i = 8\,\mathrm{m}$, though they stayed just above the safe distance $s_i = 5\,\mathrm{m}$. The velocity presented in Fig. 6.7 follows the trend set in the previous experiment, shown in Fig. 6.4, in the sense that the maximum linear velocity $v_{47max} = 1\,\mathrm{m\,s^{-1}}$ is often exceeded by UAV 47 and that the velocity is noticeably unstable. The velocity of UAV 47 reached its maximum of $\approx 1.14\,\mathrm{m\,s^{-1}}$ about 103 seconds into the flight.

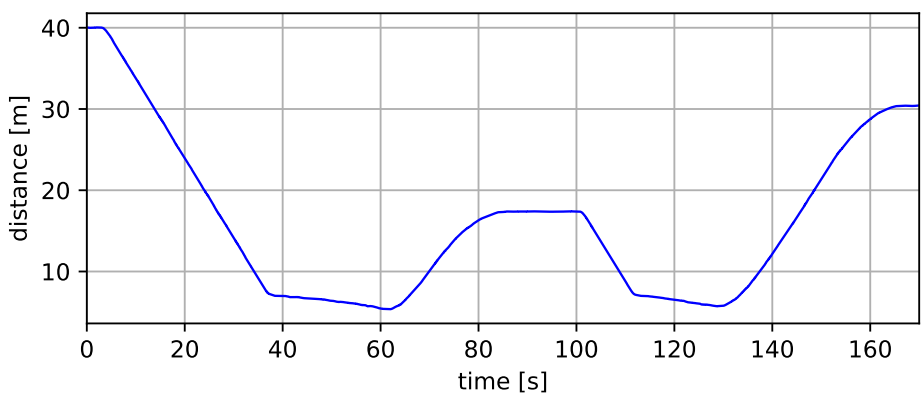


Figure 6.5: Distance between UAV 47 and UAV 60. Maximum linear velocity v_{imax} is set to $1\,\mathrm{m\,s^{-1}}$.

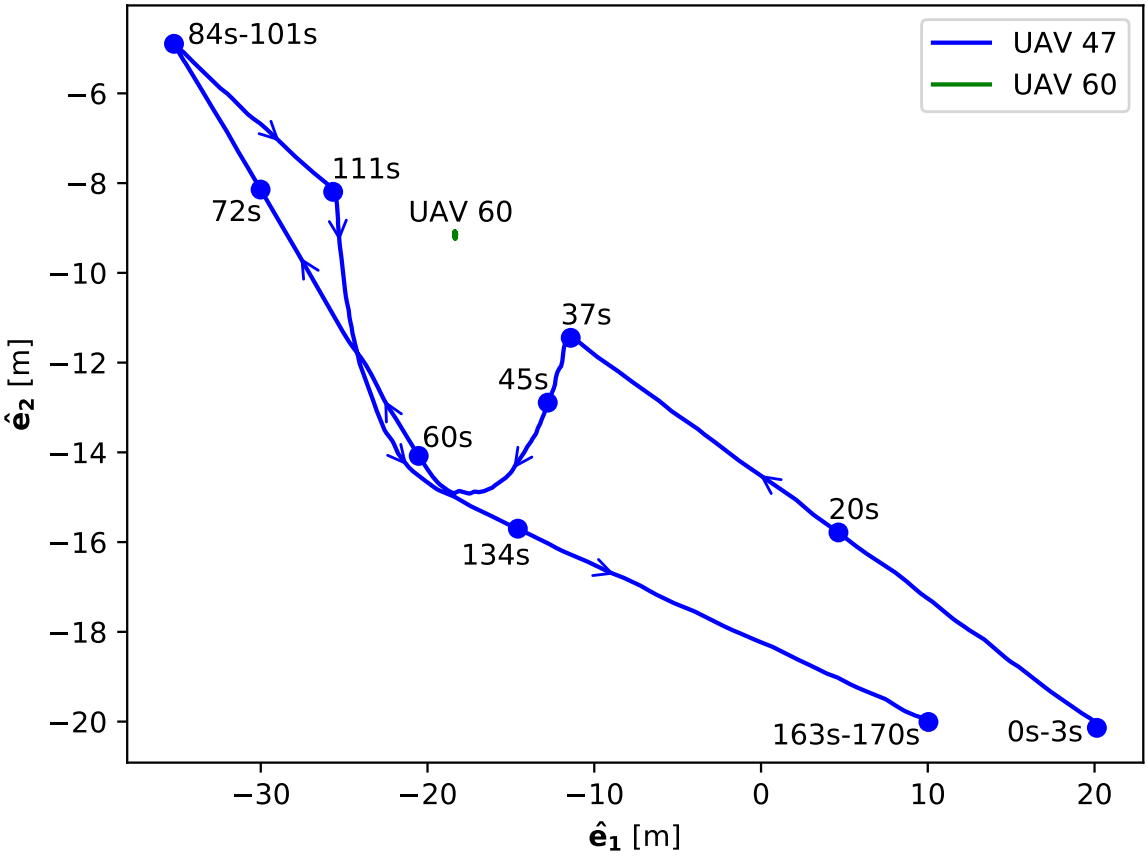


Figure 6.6: Top-down view of the trajectories of the UAVs using the proposed controller. UAV 60 is stationary, UAV 47 has to circle UAV 60 to reach the target. Maximum linear velocity v_{imax} is set to $1\,\mathrm{m\,s^{-1}}$.

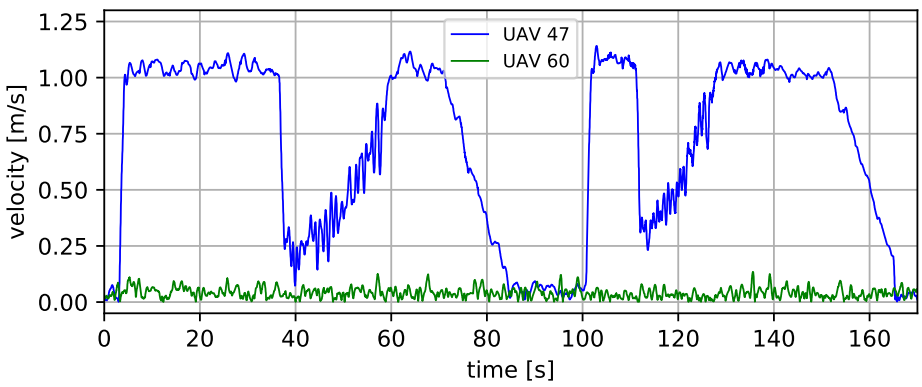


Figure 6.7: Velocities of the UAVs. Maximum linear velocity v_{imax} is set to $1\,\mathrm{m\,s^{-1}}$.

6.3 UAV avoidance at higher velocity

Because the distance between UAV 47 and UAV 60 in the previous experiment, shown in Fig. 6.5, almost reached the safe distance s_i of 5 meters, we ran a similar experiment with a higher maximum linear velocity $v_{imax} = 2\,\mathrm{m\,s^{-1}}$. In this experiment, UAV 47 was stationary.

UAV 60 circled UAV 47 to reach target $T(60)$. When UAV 60 reached $T(60)$, the target was moved to the starting position of UAV 60. The resulting trajectory is in Fig. 6.8.

The trajectory is similar to the trajectory in the previous experiment, shown in Fig. 6.6. UAV 60 successfully avoids UAV 47 in both directions without difficulties. The distance between the UAVs, shown in Fig. 6.9, remains barely above the safe distance $s_i = 5\,\mathrm{m}$. Fig. 6.10 shows that about 34 seconds into the flight, UAV 60's velocity v_{60} reached its maximum of $\approx 2.31\,\mathrm{m\,s^{-1}}$, corresponding to $\approx 115.5\,\%$ of the maximum linear velocity v_{60max}, which is comparable to the maximum of $\approx 114\,\%$ of the maximum linear velocity reached by UAV 47 in the previous experiment.

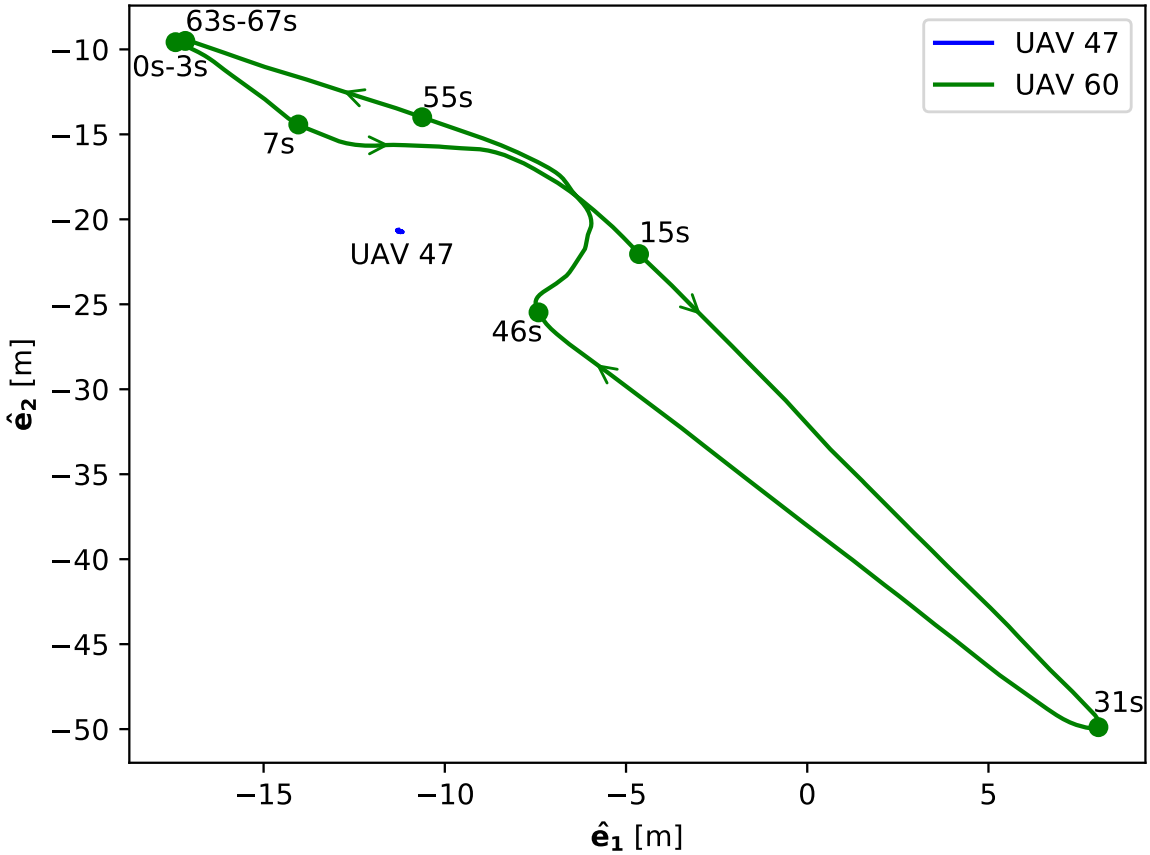


Figure 6.8: Top-down view of the trajectories of the UAVs using the proposed controller. UAV 47 is stationary, UAV 60 circles UAV 47 to reach the target. Maximum linear velocity v_{imax} is set to $2\,\mathrm{m\,s^{-1}}$.

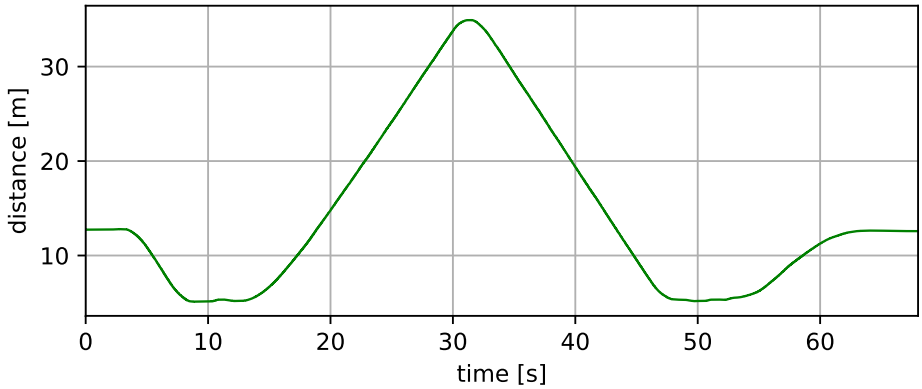


Figure 6.9: Distance between UAV 60 and UAV 47. Maximum linear velocity v_{imax} is set to $2\,\mathrm{m\,s^{-1}}$.

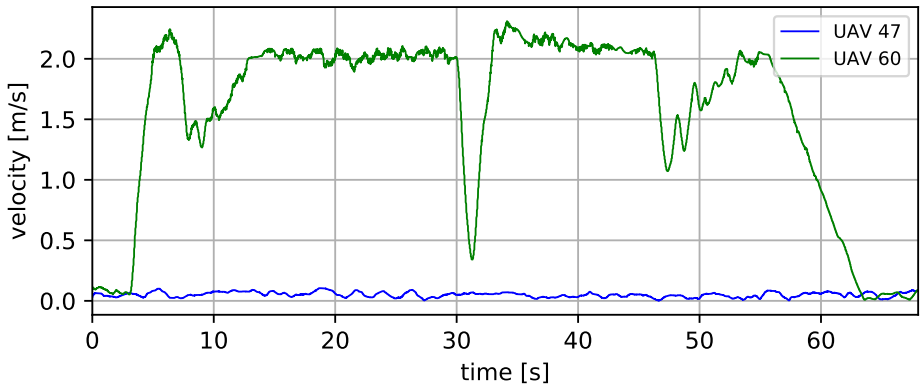


Figure 6.10: Velocities of the UAVs. Maximum linear velocity v_{imax} is set to $2\,\mathrm{m\,s^{-1}}$.

6.4 Human-swarm interaction

The last experiment was conducted to verify the workings of human-swarm interaction. The swarm consisted of three UAVs; their maximum linear velocity v_{imax} was set to $1\,\mathrm{m\,s^{-1}}$. A few seconds after the UAVs took off, a human approached them in a way that the human would be in sight of the UAVs' cameras. Afterward, the human walked around the swarm to check if the UAVs adjusted the real heading to keep observing the human. Then, about 21 seconds into the flight, the human showed the gesture Left. When the swarm started moving, the human stopped showing the gesture and followed the swarm. A few seconds after the UAVs reached their targets, about 50 seconds into the flight, the human showed the gesture Forward. Once the UAVs started moving, the human stopped showing the gesture and walked away from the swarm. The UAVs' and the human' trajectories are shown in Fig. 6.11. The trajectory of the human is shown according to the perception of UAV 47. The perception of the human by the other UAVs was nearly identical to the perception by UAV 47, except for some inaccuracies at the end of the experiment (65 seconds and further). The yaw angle $H_r(i)$ of the human is equal to zero when showing both gestures. When executing the last command, the UAVs were manually stopped by a human operator before they reached their target.

Therefore, the UAVs did not travel the entire distance $d_f = 10\,\mathrm{m}$. The detected gestures as seen by UAV 47 are presented in Fig. 6.15. A video from the experiment is available online at `http://mrs.felk.cvut.cz/prazak-2022-bp`.

As shown in Fig. 6.11, the UAVs reacted correctly to both gestures, though it took them a bit longer to evaluate the second gesture. Apart from the interval between 65 s and 68 s, the trajectory of the human is smooth and continuous, which suggests that the UAVs did not lose sight of the human and were able to observe the human's movement throughout the flight. The real heading of the UAVs, shown in Fig. 6.12, confirms that the UAVs successfully followed the movement of the human. The rapid changes in the human's trajectory occurring between 65 s and 68 s are most likely measurement inaccuracies caused by the UAVs' abrupt stop when a human operator manually ended their flight. From Fig. 6.14, we can see that the UAVs maintained the distances between one another at roughly the original value. Noticeable deviations occur when the UAVs start executing a command, which is likely caused by each UAV reaching a consensus on the perceived gesture at a slightly different time. However, because all the UAVs travel the same distance, the distance to other UAVs eventually levels out at the original value as the UAVs reach their targets, which can also be seen in Fig. 6.14. As shown in Fig. 6.13, the velocities of the UAVs behave similarly as in the previous experiments, i.e., the velocities significantly exceed the maximum linear velocity v_{imax} and are greatly unstable.

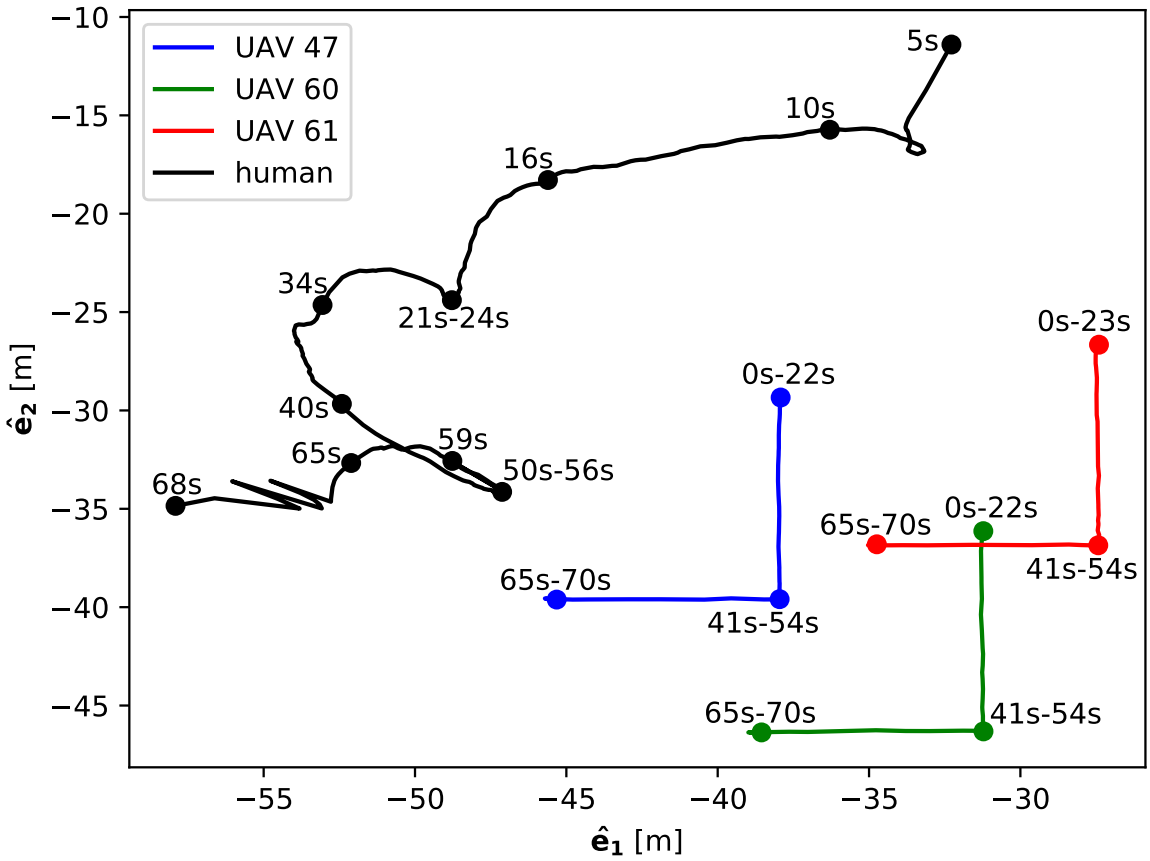


Figure 6.11: Top-down view of the trajectories of the UAVs and the human. The trajectory of the human is shown as perceived by UAV 47. A video from the experiment is available at `http://mrs.felk.cvut.cz/prazak-2022-bp`.

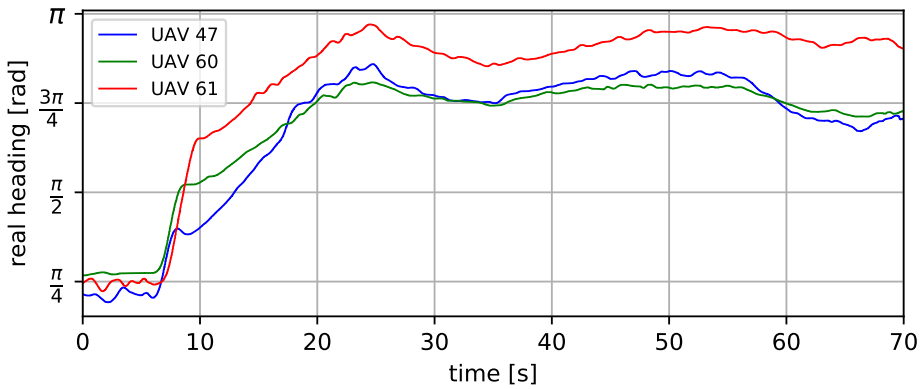


Figure 6.12: Real heading of the UAVs during the human-swarm interaction experiment.

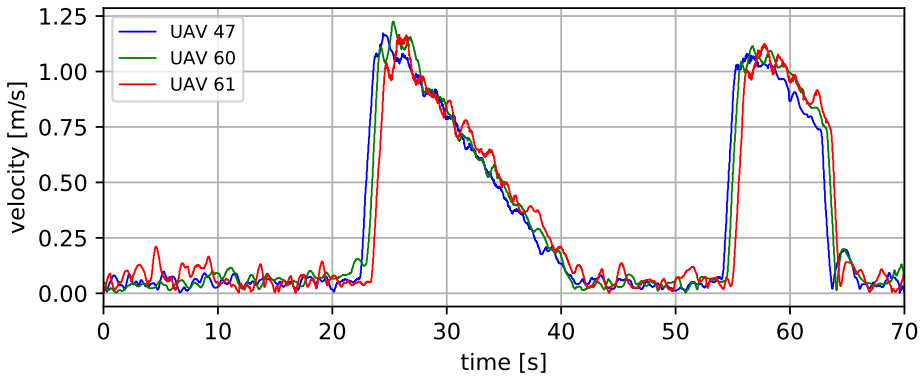


Figure 6.13: Velocities of the UAVs during the human-swarm interaction experiment.

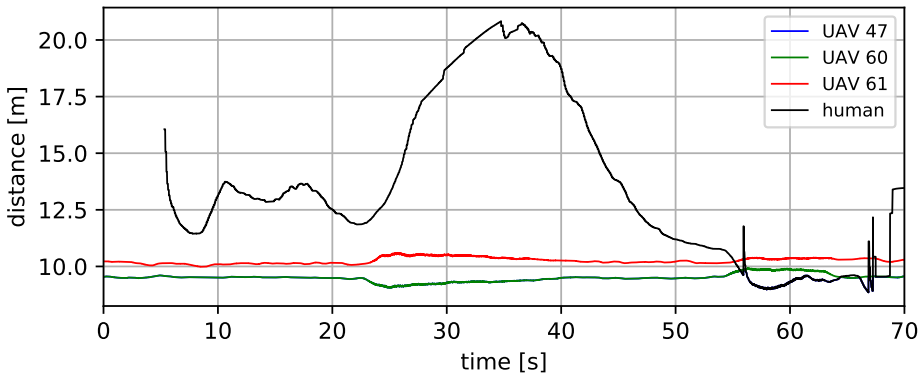


Figure 6.14: Distance to closest UAV or human during the human-swarm interaction experiment. The position of the human is taken according to UAV 47's perception.

(a) Left gesture.

(b) Forward gesture.

Figure 6.15: Gestures perceived by UAV 47 during the real-world experiment. The images on the left-hand side show raw images from UAV 47's Intel RealSense camera. The images on the right-hand side show intermediate output from the gesture estimator depicting found keypoints and connections.

Chapter 7

Conclusion

We proposed a flocking controller for swarms of UAVs derived from a flocking controller for unicycle UGVs. The migration from UGVs to UAVs required multiple changes as the abilities of the two types of vehicles differ. Notably, UAVs can move along the vertical axis. Although we did not use the vertical axis directly, we extended the sensing capabilities of the agents to consider the third dimension by using a detection spherical sector instead of a detection circular sector. Moreover, unlike unicycle UGVs, UAVs can freely change the direction of travel regardless of their real heading, which we utilized to enhance the movement of the UAVs.

Further, we designed gesture-based human-swarm interaction behavior and integrated it with the proposed flocking controller. The human's gestures represented commands for the swarm to fly in a particular direction. We equipped the UAVs with RGBD cameras and used the cameras to monitor the human's position and gestures. The UAVs then utilized a consensus algorithm to interpret the gestures and execute the corresponding commands simultaneously. Because the UAVs' real heading was independent of the UAVs' direction of travel, the UAVs could track the human and register new commands midflight. Additionally, we implemented collision avoidance with the human to make the proposed system safe to use.

Numerous experiments were run in the realistic Gazebo simulator to test the proposed flocking controller and human-swarm interaction behavior. The proposed flocking controller was also compared to a Boids-inspired flocking controller designed by the Multi-robot Systems Group (MRS) at Czech Technical University (Czech Technical University (CTU)) in Prague. Additionally, we tested the proposed flocking controller in combination with the novel UVDAR technology for relative localization to illustrate the controller's properties in environments where GNSS might be unavailable; however, the results from the experiment showed that future work might be necessary in this regard. Finally, we conducted several hardware experiments that verified the proposed system's correctness in the real world.

Future work might look into some of the drawbacks of the current design and implementation of the flocking controller. Sec. 5.4 identified various problems when using the proposed flocking controller with the novel UVDAR relative localization technology. Because GNSS localization is not always available, successful integration of relative localization using UVDAR or a similar system would be necessary for the proposed flocking controller to work reliably in indoor or otherwise GNSS-denied environments. During the hardware experiments, we discovered that the velocities of the UAVs often exceeded the predefined maximum velocity and were significantly unstable, which is a phenomenon not seen when testing the proposed controller in the Gazebo simulator. Such non-deterministic behavior could be problematic in real-world applications. Lastly, the system could be expanded to include avoidance of general obstacles.

Chapter 8

References

[1] A. Ahmad, V. Walter, P. Petráček, M. Petrlík, T. Báča, D. Žaitlík, and M. Saska, "Autonomous Aerial Swarming in GNSS-denied Environments with High Obstacle Density," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 570–576. DOI: 10.1109/ICRA48506.2021.9561284.

[2] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "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. 1, pp. 1–28, 2021. DOI: 10.1007/s10846-021-01383-5.

[3] G. A. Cardona and J. M. Calderon, "Robot Swarm Navigation and Victim Detection Using Rendezvous Consensus in Search and Rescue Operations," *Applied Sciences*, vol. 9, no. 8, 2019, ISSN: 2076-3417. DOI: 10.3390/app9081702.

[4] D. Shah and L. Vachhani, "Swarm Aggregation Without Communication and Global Positioning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 886–893, 2019. DOI: 10.1109/LRA.2019.2893413.

[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," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2637–2644, 2019. DOI: 10.1109/LRA.2019.2901683.

[6] 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)*, IEEE, pp. 1217–1226. DOI: 10.1109/ICUAS.2018.8453331.

[7] V. Walter, N. Staub, M. Saska, and A. Franchi, "Mutual Localization of UAVs based on Blinking Ultraviolet Markers and 3D Time-Position Hough Transform," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 298–303. DOI: 10.1109/COASE.2018.8560384.

[8] B. Wang, S. Li, J. Guo, and Q. Chen, "Car-like mobile robot path planning in rough terrain using multi-objective particle swarm optimization algorithm," *Neurocomputing*, vol. 282, pp. 42–51, 2018, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.12.015.

[9] J. Nagi, A. Giusti, L. M. Gambardella, and G. A. Di Caro, "Human-Swarm Interaction Using Spatial Gestures," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3834–3841. DOI: 10.1109/IROS.2014.6943101.

[10] A. Kolling, K. Sycara, S. Nunnally, and M. Lewis, "Human-Swarm Interaction: An Experimental Study of Two Types of Interaction with Foraging Swarms," *Journal of Human-Robot Interaction*, vol. 2, no. 2, pp. 104–129, 2013. DOI: 10.5898/JHRI.2.2.Kolling.

[11] T. Schmickl and H. Hamann, "BEECLUST: A Swarm Algorithm Derived from Honeybees," *Bio-Inspired Computing and Networking. CRC Press (March 2011)*, 2011.

[12] C. Vasile, A. Pavel, and C. Buiu, "Integrating human swarm interaction in a distributed robotic control system," in *2011 IEEE International Conference on Automation Science and Engineering*, pp. 743–748. DOI: 10.1109/CASE.2011.6042493.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, Kobe, Japan, vol. 3, 2009, p. 5.

[14] S. Bashyal and G. K. Venayagamoorthy, "Human Swarm Interaction for Radiation Source Search and Localization," in *2008 IEEE Swarm Intelligence Symposium*, pp. 1–8. DOI: 10.1109/SIS.2008.4668287.

[15] D. Miner and M. desJardins, "Learning Abstract Properties of Swarm Systems," in *Proc. 8th International Conference on Autonomous Agents and Multiagent Systems*, Citeseer, Jan. 2008.

[16] R. Poli, "An Analysis of Publications on Particle Swarm Optimization Applications," *Essex, UK: Department of Computer Science, University of Essex*, 2007.

[17] C. Hartman and B. Beneš, "Autonomous Boids," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 199–206, 2006. DOI: 10.1002/cav.123.

[18] M. Reyes-Sierra, C. C. Coello, *et al.*, "Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006, ISSN: 0973-1873.

[19] E. H. Gustafson, C. T. Lollini, B. E. Bishop, and C. E. Wick, "Swarm Technology for Search and Rescue Through Multi-Sensor Multi-Viewpoint Target Identification," in *Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 2005. SSST'05.*, IEEE, pp. 352–356. DOI: 10.1109/SSST.2005.1460935.

[20] Y. Shi and R. C. Eberhart, "Particle Swarm Optimization: Developments, Applications and Resources," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, vol. 1, 2001, pp. 81–86. DOI: 10.1109/CEC.2001.934374.

[21] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," in *Proceedings of ICNN'95-International Conference on Neural Networks*, IEEE, vol. 4, 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[22] 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, 25–34, ISBN: 0897912276. DOI: 10.1145/37401.37406.