Politecnico di Torino

DEPARTMENT OF CONTROL AND COMPUTER ENGINEERING (DAUIN)
Master of Science in Mechatronic Engineering

Master of Science Thesis

Fully autonomous flocking behavior of flapping wing robots

Thesis Advisor:
Prof. Alessandro Rizzo, Politecnico di Torino
Prof. Guido C.H.E. de Croon, Delft Technical University

Candidate:
Veronica Munaro

Academic Year 2020-2021

Abstract

The present work proposes a solution for performing indoor flocking with micro air vehicles (MAVs) in a fully autonomous way, considering the motion of the robots in three dimensions. The main characteristics that are looked for in the behavior of the flock are: adaptability, scalability and robustness.
In order to meet these goals, each agent is equipped with on-board sensors that allow it to navigate autonomously in the environment; specifically, the IMU provides information about the velocity and acceleration of the drone and about its pose, while an UWB antenna measures the relative distance from every other agent in the flock and broadcasts the information extracted by the IMU to the rest of the flock. Basing on these measurements only, a state estimator allows each agent to obtain accurate relative localization of any other agent within UWB range of work. The estimation is done by means of an Extended Kalman Filter, which outputs both the relative position and the relative yaw angle of a MAVj (tracked) with respect to a MAVi (host). The resulting system does not rely on any external infrastructure, such as UWB beacons or the GNNS, making the flock deployable anywhere at any time.
The limitations of this solution are studied through an observability analysis in Lie derivatives, which leads to the identification of conditions in terms of combinations of inputs and states, that would affect the convergence of the filter.
Finally, two flocking algorithms are presented, inspired by the Reynolds' theory of boids' rules of alignment, cohesion and separation.
Emphasis is given to the implementation of a completely decentralized control, such that the resulting system appears highly robust to the loss of one or more agent and at the same time scalable to changes in the size of the flock, whose members can be easily added or removed without changing anything in the algorithm. The overall collective motion that emerges is the result of purely local interaction, without the need of a preprogrammed path or formation control. This feature makes the flock indipendent from human guidance and able to adapt to any environment or situation dynamically.
The EKF and the flocking algorithms are coded in MATLAB & Simulink in discrete time and integrated with the preexisting model of a flapping wing MAV, the DelFly Nimble, designed and developed at the Technical University of Delft.

Estratto in lingua italiana

Il lavoro svolto per questa tesi mira a proporre una soluzione per implementare uno stormo di micro veicoli aerei (MAVs abbreviato) nell'ambito della robotica degli sciami, completamente autonomo e adatto sia per applicazioni indoor sia outdoor. Le caratteristiche fondamentali che si ricercano sono: adattabilità, scalabilità e affidabilità.
A tal fine, ogni agente nel team è dotato di sensori di bordo che gli permettono di navigare autonomamente nell'ambiente circostante; in particolare, il Sistema di Navigazione Inerziale (a cui si fa riferimento con la sigla inglese IMU nel seguito) permette di ottenere informazioni sulle velocità e le accelerazioni del drone e sulla sua orientazione nello spazio, mentre un'antenna Ultra Wide Band (UWB) è in grado di misurare la distanza del robot dagli altri membri del gruppo e di diffondere le misurazioni effettuate dall'IMU. Basandosi esclusivamente su tali informazioni, un osservatore di stato consente a ciascun agente di stimare accuratamente la posizione relativa di ogni altro membro dello stormo entro il range permesso dall'antenna UWB. La stima è effettuata attraverso un Filtro di Kalman Esteso (abbreviato EKF), che ha lo scopo di fornire sia la positione relativa sia l'angolo di imbardata relativo del MAVj (osservato) rispetto al MAVi. Il sistema risultante appare indipendente da qualsiasi infrastruttura esterna, come ancore UWB o GPS, rendendolo adatto ad essere implementato in qualsiasi contesto.
I limiti di tale soluzione sono studiati attraverso un'analisi di osservabilità nelle derivate di Lie, che permette di identificare specifiche condizioni in termini di input e stati che possono potenzialmente influenzare negativamente la convergenza del filtro.
Infine, sono stati sviluppati due algoritmi per emulare i reali stormi di uccelli o sciami di insetti, basandosi sulle tre leggi fondamentali derivate nella teoria di Reynolds' dei "boids", entità dotate di una determinata velocità in termini di modulo e direzione che aspirano a simulare il moto degli agenti in un gruppo che presenta le caratteristiche di uno sciame: allineamento delle velocità, coesione e elusione delle collisioni.
L'aspetto a cui viene data maggiore importanza è la realizzazione di un sistema di controllo interamente decentralizzato, in modo tale da ottenere un sistema che sia robusto alla perdita di uno o più agenti e allo stesso tempo incrementabile dal punto di vista della dimensione dello stormo, a cui è possibile aggiungere o rimuovere membri senza modificare significativamente l'algoritmo. Il comportamento collettivo globale emergente è il risultato di interazioni esclusivamente locali, senza la necessità di seguire una traiettoria preimpostata. Per questa ragione, il sistema può essere implementato in assenza di un pilota umano ed è in grado di adattarsi dinamicamente all'ambiente.
Il Filtro di Kalman Esteso e l'algoritmo per simulare gli sciami sono stati scritti in ambiente MATLAB/Simulink e integrati con il modello del DelFly Nimble, un drone ispirato alle

mosche della frutta che sfrutta il volo battuto, sviluppato dall'Università Tecnica di Delft (TUDelft).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The present work aims at proposing a new fully on-board, range-only localization algorithm for the estimation of the relative position in the three dimensions of MAVs, to enable a flocking behavior of flapping-wing drones. The project is intended for the application to the DelFly robotic platform developed by the MAVLab, TUDelft, for performing indoor entertainment shows.
The thesis is structured as follows:

Chapter 2: An overview on the theoretical background is given, presenting the flocking theory of boids, a review of the state of the art in the field of swarm robotics, an introduction to state estimators for nonlinear systems and a mathematical description of the concept of observability, with emphasis on the approach with Lie derivatives for the study of nonlinear systems

Chapter 3: The problem of range-only relative position estimation is introduced and the Extended Kalman Filter designed for solving it is explained, followed by a description of the simulation environment built to test the estimator. The obtained results are shown and a preliminary evaluation of the filter's perfomance is proposed.

Chapter 4: The chapter is dedicated to the observability analysis with Lie derivatives. The mathematical computation of the observability matrix and of its determinant is carried out and the degree of observability of the system is studied with different combinations of inputs and states through the local weak observability index C^{-1}. The relevant cases are analyzed more in depth and proof of the correctness of the approach is given by tests in simulation. Finally, the EKF estimation quality is evaluated on the basis of the results obtained through the observability analysis.

Chapter 5: Two flocking algorithms are outlined, the first designed as a purely emergent behavior dictated by local interactions, able to operate in a constrained environment, the other obtained as a leader-follower task with collision avoidance within the flock and velocity matching.

Chapter 6: The chapter summarizes the results obtained and proposes a comparison with the 2D case, highlighting the main differences between the two. Suggestions for further studies on the basis of the results in the present work are proposed.

1.1 Motivation and context

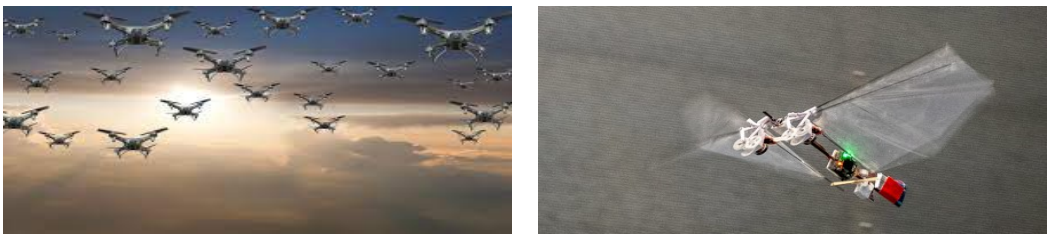
When we happen to observe the murmuration of starlings twisting in beautiful clouds in the sky or herds of wildlife traversing the African savannah during the Great Migration, a few questions pop up: how do they do that? How can groups of hundreds, or thousands, of unwitting animals coordinate their motion in such a skillful manner? The mystery gets even thicker when we observe the same behavior in crowds of humans performing in a flash mob or in a concert, when, unaware of the process that is occurring, we somehow find ourselves getting the moves with our neighbor or raising with perfect timing in a "ola".

(a) *Shoal of fish* **(b)** *Flock of birds*

Figure 1.1: *Examples of flocking in nature*

This fascinating behaviour is inexplicable, and yet very common in nature, so much that it bewitched researchers in different fields of science for decades, if not centuries, and it has not been fully understood yet. The advantages driven by the species that manifest this attitude, referred to as flocking behavior, are undeniable: cooperation can allow a faster and more efficient completion of tasks, coordination can ensure a better coverage of a foraging area and harmonization can make the flock more robust to the attacks of predators.
It is not surprising, then, that the idea of exploiting such behavior in human applications can be tempting. Imagine armies of rovers exploring the Mars surface without the need for human guidance or any networked infrastructure, or flocks of Micro Air Vehicles making their way in wrecked environments, securing the path for the incoming rescue team. The advantages would be stunning. For this reason, in recent years various branches of science have developed an interest for such concepts like swarming and flocking.
The first attempt to emulate this behavior can be found in the theory developed by C. Reynolds in 1987 ([34]), known as the *theory of boids*: it introduces the concept of "boid", an entity that can represent an imaginary animal, and it states that the ability of animals to move as a single entity can be reconducted to the application of three simple rules, namely alignment, cohesion and separation. The observation of neighbors triggers a response in each individual, that adjusts its own velocity according to these three driving needs, producing the incredible effect that we can observe.
Since then, Reynolds' theory, originally intended for graphic simulation, has provided the bases to a new branch of engineering, known as "Swarm Robotics".
It this context, many approaches to the task have been put forward, ranging from formation control, where agents occupy specific relative positions, to consesus algorithms for selecting collectively a preferred trajectory.
Flocking fits into the background of swarm robotics, providing an example of self-organized behavior, produced solely by the dynamic interaction of each member with its neighbors.

The advantage of this solution compared to the others is that this emergent behavior is a consequence of purely local interactions within the group. For this reason, it could be advantageously exploited to produce fully autonomous flocks, able to perform in any situation indipendently of the environment. Very few examples of infrastructure-indipendent swarms are available in literature, as the most common solutions available today usually rely on external localization systems like Global Navigation Satellite System (GNSS) or take advantage of beacons strategically located in the area (see [6], [38] for a complete review of current applications).
Very few examples exist which rely on only on-board equipment and range-only measurement for 3D problems. An important work in this direction has been carried out by Trawny in [33] and [32], where the authors aim at finding a solution for the 6 degree of freedom (d.o.f.) relative-pose estimation problem in 3D by using different combinations of limited information, like robot-to-robot distance measurement and dead-reckoning, bearing-only measurement, bearing and distance measurement and distance-only measurement and finding the minimum amount of measurements required to perform the estimation. The work is completed by a nonlinear local weak observability analysis to find sufficient conditions for the 3D relative pose to become locally weakly observable and by a nonlinear weighted least squares estimator to validate the algorithm under different amounts of noise.
The present work aims at filling this gap, providing a novel solution for performing a fully autonomous flock of MAVs basing on a range-only relative localization procedure in the three-dimensional space and deploying the designed algorithm onto the most exotic platform provided by the DelFly Nimble.



(a) **(b)**

Figure 1.2: *Swarm of UAVs (a) The DelFly Nimble (b)*

1.2 Contribution of the present work

Making a step forward from the previous work [26], which aims at developing an estimator for performing relative-localization in 2D using only range-mesurement, with this thesis we try to move beyond the two-dimensional setting by adding the third dimension. The drones are now moving in 3D and able to guess the relative position of the other agents by estimating four states: the position $\mathbf{p}_{ij}$ in the three directions and the relative heading ψ_{ij}. The real altitude at which each robot is flying is considered unknown, although it could be estimated by a laser range: the reason for this choice is to produce a system that could operate indifferently on an uneven ground or at relevant heights and which is not influenced by the orientation of the drone, that could affect the measurement of the distance especially

in the case of flappers, which are able to make quick turns and change their roll angle very sharply. The resulting system is expected to satisfy the fundamental requirements of collective robotics of robustness, flexibility and scalability by exploiting a modular structure of the flock, so that each robot results fully capable of navigating autonomously, relying only on on-board equipment.
The available sensory knowledge is provided by an UWB antenna mounted on each drone, which fulfills the role of measuring the range from all the other members and broadcasting the information on velocity and attitude as produced by the IMU. These measurements are input to a state observer, which carries out the estimation of the relative position. The Extended Kalman Filter is chosen as estimator because it seems to be the best fit for the system model considered and its low processing and memory requirements make it suitable for working on-board of small MAVs with limited computational capacity.
In order to quantify the EKF performance, an observability analysis is executed in terms of Lie derivatives. To evaluate the observability of a system means to understand how good the filter is at providing an univocal output when a certain input is applied and can be done by studying the observability matrix $\mathbf{O}$ of the system.
For nonlinear systems, like the one considered in this study, the observability matrix can be built by computing the Lie derivatives of the measurement, the range $\mathbf{h}$ in this case. Once $\mathbf{O}$ has been built, some specific indeces can be adopted to evaluate not only *if* the system is observable, but also *how well* it is observable. This kind of study has proved to be very useful to understand what to expect from the filter and to identify possible manouvres that could make the estimation unreliable.
The results obtained from this stage are exploited to design the flocking algorithms, that should dictate the trajectory followed by each agent basing on the estimated position computed by the EKF. The algorithm plays the role of a controller by computing the desired velocity for each MAV at every time step in such a way that a collective emergent behavior can be obtained. Two algorithms have been designed, both suited for indoor applications, as well as for outdoor ones.
The first one is inspired by [43] and [41], where a flock simulation is carried out in 2D by exploiting the alignment rule and the collision avoidance one; moreover, the algorithm includes also a smart solution for embedding a wall avoidance feature by employing the concept of "shill agents", mock agents that are used to keep the flock within a bounded arena. A self-propelling term is also added, which allows the boids to keep moving even in the case where they are separated from the flock. In this thesis, the algorithm is modified to be applied in three dimensions and integrated with the relative-position estimator and the real model of the drone. The drones are restricted to a box that can be interpreted ideally as a room, and the components that contribute to the computation of the target velocity are weighted, drawing inspiration from the Cucker approach [9], that bases its model on the postulation that each agent in the flock adjusts its velocity as the weighted average of the difference of its velocity with that of its neighbors.
The second algorithm is based, instead, on the Couzin model ([8]) and associates the implementation of Reynolds' rules for keeping the swarm cohesive and safe with a leader-follower task. The advantage of this solution is the possibility of controlling the trajectory of the flock by imposing a path to one informed agent. The remaining of the flock is able to travel of a similar route thanks to a communication protocol, that propagates the information in cascade through the flock. This solution relies more heavily on the ability to correctly localize the other members of the flock compared to the previous one, as the relative position of the preceding robot becomes a weighted component in the computation of th target velocity of each drone.

The significant advantage of these solutions is the possibility of achieving a coordinated motion of the robots by relying on a minimum amount of equipment that can be easily carried on-board even by the smallest MAVs.

Chapter 2

Theoretical background

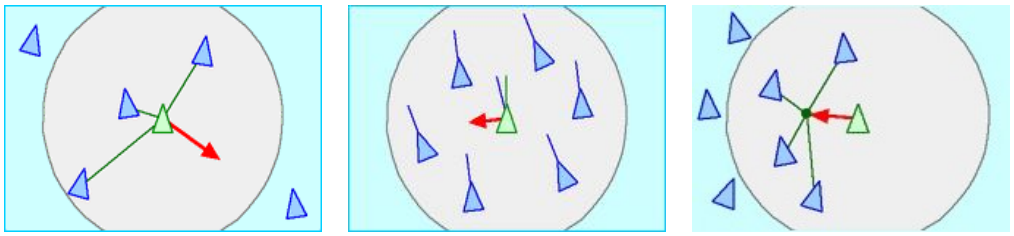
2.1 Theory of boids and flocking behavior

Flocking is a collective behavior that can be observed in many species, ranging from birds to fish and insects. It has been traditionally object of study in the biological framework and has recently gained popularity in other domains, like computer simulations and engineering. The earliest attempt to emulate the flocking behavior observed in natural systems can be found in the work by C. Reynolds [34] 1987, in which a computational model of flocking is put forward. The theory is known as the theory of *boids* and set the path to this new field of research.
The main focus of Reynolds' work is on the computer simulation of flocks of birds and has been applied in videogames and by the movie industry in productions like *Batman Returns* (1992) and *The Lion King* (1994) for the generation of the wildbeest stampede and the bat swarm respectively, but the theory can be equally applied to the modeling of schools and herds, as a *boid* is generically defined as an artificial bird (or entity) with a an individual velocity and direction of motion that is able interact with the other surrounding agents.
The theory of boids appeared at once to be significantly appealing and soon gained popularity thanks to its remarkable simplicity: its main finding is that the behavior of flocks can be simulated with reasonable resemblance with natural networks by relying only on very basic, yet fundamental rules. According to Reynolds' [34], the collective motion is the result of each agent's tendency to obey to the following laws:

- Cohesion: the desire to stay in the group
- Separation: the necessity of avoiding collisions with the neighbors
- Alignment: the inclination to match velocities with nearby flockmates

Basing on these three driving forces, each boid computes its own target velocity in terms of direction and magnitude, obtaining a good similarity with natural flocks.

The three drives affect the behavior of the flock in a different manner, as they are computed relying on different information about the sorrounding agents, and they can therefore be classified as *static* and *dynamic* [25]: separation and cohesion are referred to as static, since they only require knowledge on the relative position of the other boids and lead to a certain equilibrium of the system, while velocity matching is labeled as dynamic because



(a) *Separation* (b) *Alignment* (c) *Cohesion*

Figure 2.1: *Reynold's theory of boids*

it is based only on the comparison of velocities in terms of direction and magnitude, without considering the relative position. If we tried to simulate a flocking behavior without the alignment component, we could observe that the agents will tend to occupy a certain position in the space and to progressively decrease their velocities to zero as they reach an equilibrium point given by the concurrent application of repulsion and cohesion.
Cohesion, often referred to as flock centering, models the agent's urge of staying within the flock. It can be seen as the need of the agent to be evenly surrounded by other boids by occupying the center of the group formed by the nearby flockmates within its field of view. This allows the flock to be compact and cohesive.
On the other hand, separation and velocity matching have the purpose of providing stability and safety to the flock: once the agents have come together thanks to the cohesion driving force, separation allows the members to keep a "living space" around themseves, which results into a number of spheres centered on each agent that other boids cannot enter; velocity matching can, instead, be thought as a predictive way of avoiding collisions, since once the agents have found a suitable equilibrium between cohesion and separation, the alignment of velocities helps to maintain it [25]. This is the reason why some theories have been developed which do not require the application of all of the rules: velocity matching proves often enough to guarantee stability and cohesion of the flock by allowing to select a common direction of motion for all the members.
One example is the Vicsek's model ([42]), a pillar in the field of flocking simulation, which focuses only on alignment of the velocities. The model drives inspiration from the orientation of particles in ferromagnetic materials in the electromagnetism framework and obtains motion of the particle between two time steps by applying only this concept. The system is made-up by self-driven particles, which adjust their headings and acceleration as the average of the neighbors' ones within a certain radius r between two time-steps. The resulting model appears to show realistic dynamics, despite its intrinsic simplicity.
Another relevant instance of reasonable simulation without employing all of the rules is introduced by Cucker and Smale in [9]: the model initially proposed a solution based on the assumption that each agent dynamically adjusts its own heading and velocity by computing a weighted average of the differences of its own velocity and the velocities of the other members within sight. The theory was later modified by adding a collision avoidance features, that pre-empts hits within the group ([10]).
However, the best-known and widely applied model that exploits communication within the flock limited to a certain volume around the agent has been proposed by Couzin in 2002 [8] and has been dictated by an attentive attempt to study and understand the behavior of natural systems. The Couzin model has inspired numerous works, ranging from robotic ap-

plications to biological studies, and is based on the idea of providing a model to understand the leadership mechanism and decision-making in animal groups. The theory introduces the idea that flocks may be made up by heterogenous members from the behavioral point of view, as it distinguishes between gregarious agents, which only apply the three main rules for flocking [34], and informed agents, which add a will term that can be weighted and represents their tendency to follow a preferred trajectory. The percentage of informed and gregarious agents needed to follow the imposed path depends on the overall size of the flock in terms of number of entities and it affects the successfull tracking of the imposed trajectory.

In the previous discussion there have been numerous references to the concept of *neighbours*: this is an important point to be considered, which is linked to the criterion according to which the agents choose how many and which other flockmates to interact with and to the communication protocol that regulates the information diffusion within the swarm. It has been proved ([34]) that birds do not consider all the members of the flock to compute their own motion, but only a small subset of it. Many approaches have been put forward that hypotize a *topological* interaction, that exploits communication with the M closest agents independently by their distance, or a *range-based* interaction like the Couzin model [8], where the liasion takes place with all the members within a certain distance or that fall within a given angle of sight ("field of view" of the agent); for these reasons, it is important to define the concept of "nearby flockmates" when addressing a flocking problem, since the elected communication protocol will affect the overall behavior of the system.

To summarize, the main foundation of Reynolds' theory is that the emergent behavior obtained with the application of these three basic rules takes the traits of a self-organizing operation and it is the result of purely *local* interactions, without the need for a complex centralized superstructure. The implications of this assumption are that the the boid has a local perception of the world within a given range or field of view that results into a number of perception spheres corresponding to each boid. It is important to stress this concept since it provides the main motivation for applying flocking algorithms to engineering problems. It is, in fact, the decentralized nature of such solutions that provides the most relevant advantages for developing fully scalable and robust systems. The expression of such field of research is in swarm robotics, a new and auspicious branch of engineering that aims at addressing diverse engineering problems by proposing solutions based on a cooperative approach and distributed control of multiple agents.

2.2 State of the art of swarm robotics

Swarm robotics is a very promising field of research that has gained attention in the last decade thanks to the large envisioned potential, although real-world applications are still rare nowadays. The goal of this branch of engineering is to produce teams of autonomous robots that can collectively solve tasks by exploiting coordinated motion. The source of inspiration is drawn from natural behaviors of social animals, such that of birds or bees, that are able to join forces to carry out activities that would be impossible to accomplish by a single individual. The main limitation to industrial applications comes from the numerous challenges that arise when the requirements for the design of individual robots that are able to safely and autonomously navigate the environment meet those derived from the design of controllers that can allow the implementation of the swarm.

(a) (b)

Figure 2.2: *Examples of swarm robotics performed with MAVs at the GRASP Lab [24] (a) and with small wheeled robots (b)*

The common methods for performing flocking behavior exploit external equipment that can provide the positioning of the agents with respect to a common, global reference frame. For outdoor applicationa GNSS is normally employed to provide the localization data ([12],[37],[40],[14]). The main issue carried by the GNSS is the limited applicability, as it can not be used in indoor environments and GPS denied ones; moreover, it is affected by low accuracy of the positioning, which makes it less suited for employment on MAVs that need to fly tightly spaced and avoid collisions. An alternative can be found in the set up of the environment, which can be arranged with a network of transreceivers, like UWB beacons. The agents receive information about their own position in a global reference frame and about the other members locations from these devices. Some examples are provided in [44] and [30]. The drawbacks of this solution lie in the staticity of the structure, that has to be settled in advance and that limits the area of applicability to the coverage provided by the systems. On the other hand, the accuracy is increased with respect to GNSS, which makes it suited for indoor applications that require an elevated precision.
In order to obtain a more versatile solution, agents should be able to perform all the measurements needed for relative localization relying only on on-board equipment. The most immediate solution would be to employ vision through a camera mounted on-board, although this method is limited by the field of view of the instrumentation. Moreover, for platforms with higher dynamics the working of the camera can be affected by the orientation of the robot.
One solution that has been put forward recently is to rely on omnidirectional sensors based on radio signals such as UWB antennas on each robot ([13]). Each transreceiver is able to estimate the relative spacing with respect to any other robot in the flock and concurrently keep track of its own position from a starting point gaining information from on-board sensors. The problem of this solution is that, over time, these measurements may be subject to drift.
Another method put forward by [7] proposes the use of bluetooth for communicating the measurements between MAVs such as velocities, height and orientation, while the distance is estimated by considering the strength of the signal.
The drawback of these solutions is that they need to refer the localization information to a common global frame, which requires the knowledge about each MAV's heading to North, acquired through magnetometers, which are subject to drift and sensitive to local magnetic disturbances [1], especially indoor.
In order to overcome this problem, [15] proposes a solution that removes the dependency on

the heading by developing a range-based relative localization method that expoloits information from the IMU and the range measured by the UWB antennas mounted on each drone. A similar solution is then used to implement a fully-autonomous swarm of MAVs in indoor in [26]. Here, each MAV is able to accurately localize any other MAV in the swarm basing only on the measurement of the range and on the communication of velocities, orienation and yaw rates as provided by the IMU. The model can estimate only the relative position in 2D (x and y direction), while the relative height is computed by comparing the height measured by each drone with a laser ranger.
The model made up by two MAVs, where MAVi attempts to estimate the relative position of MAVj, is described as follows:

$$\dot{\mathbf{X}}_{ij} = f(\mathbf{X}_{ij}) = \begin{bmatrix} R(\psi_{ij})\mathbf{v}_j - \mathbf{v}_i - Sr_i\mathbf{p}_{ij} \\ r_j - r_i \end{bmatrix} \tag{2.1}$$

$$R(\psi_{ij}) = \begin{bmatrix} cos(\psi_{ij}) & -sin(\psi_{ij}) \\ sin(\psi_{ij}) & cos(\psi_{ij}) \end{bmatrix} \tag{2.2}$$

$$S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{2.3}$$

where ψ_{ij} is the relative yaw angle, r_i and r_j are the yaw rates, v_i and v_j are the velocities and $X_{ij} = [x_{ij}, y_{ij}, z_{ij}, \psi_{ij}]^T$ is the state vector.

The present work employs [15] and [26] as a starting point for the relative localization algorithm by extending it to 3D and by removing the information about the height. In the following sections, the relative localization algorithm described in 2.1 will be referred to as the *2D case* for making a comparison with the solution developed in this thesis.

2.3 Extended Kalman Filter

There exist situations where a system presents states that are needed to design a controller, but their value cannot be measurend directly, or where the measurement is affected by noise and therefore is unreliable and has to be obtained my merging the information from different sensors. Various examples can be found in any engineering field, like guidance and navigation systems, computer vision and signal processing. To tackle these problems, some smart solutions have been found, naming the optimal state estimators.
An optimal state estimator is able to find the best approximation of an unknown variable without measuring it directly or to merge the information coming from a number of noisy sensors to optimally estimate that variable.
One of the best known solutions is the Kalman Filter, introduced in the '60s by the Hungarian - American engineer Rudolf Emil Kalman (1930 - 2016) in [35] and [36]. It has originally been designed for linear systems, but through time new versions has been proposed to solve also nonlinear problems.
It has the form of a recursive algorithm that performs an a-priori estimation of the states and the expected error covariance and corrects it with an a-posteriori update with the knowledge about the expected values of both thanks to the measurement of some quantity. The filter provides an optimal solution from a statistical point of view and operates through these two steps (*a-priori* and *a-posteriori*) both in the linear and non-linear applications.

2.4 Kalman Filter

Kalman Filter is an optimal state estimator that addresses linear systems.
In order to explain the functioning of an optimal state estimator, let's consider a general LTI DT system described in state-space form as follows:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_k \\ y_k &= Cx_{k-1} + Du_{k-1} + v_k \end{aligned} \tag{2.4}$$

where x_k is the state, u_k is the input and y_k is the set of measurements performed by the system. The notation k indicates the time step considered and k-1 the previous step.
Since the measurements y_k are obtained through real measurement instruments, it cannot provide a perfect reading of the quantity and the output will be affected by noise; the same can be said for the possible disturbances to which the system might be subject between two time steps. For this reason, it is possible to account for such noise by adding the terms v_k and w_k, referred to, respectively, as the *measurement noise* and the *process noise*. Although these signals are random and don't follow a pattern, the most realistic and easy-to-handle description can be derived from probability theory using their average properties, by representing them as zero-mean white gaussian noise. The distribution of the disturbances can be described as follows:

$$\begin{aligned} v_k &= N(0, R) \\ w_k &= N(0, Q) \end{aligned} \tag{2.5}$$

where the 0 implies that the error due to the noise will be mostly around 0, while R and Q are the error covariance matrices. For simplicity, they are assumed to be equal to the error variance, σ_v^2 and σ_w^2. If the system has more than one state/measurement they become diagonal matrices with the error covariance of each state/measurement on the diagonal. A graphical representation of the distribution is shown in Figure 2.3:

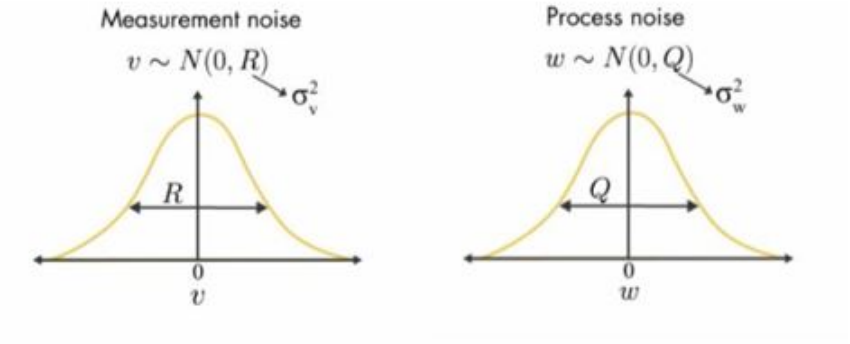


Figure 2.3: *Zero-mean white noise gaussian distribution*

Choosing the values for the covariance matrices is not trivial at all, as it can heavily affect the performance of the filter. A realistic estimation of the noise is important to guarantee

a good prediction of the state. This topic will be investigated in detail in Chapter 3, where the design of the optimal state estimator used in this thesis will be addressed.

To recap the conditions of the problem addressed by state estimators, we can recall that, due to noisy information coming from sensors the measurements are unreliable: it is therefore necessary to find a way to correct this knowledge. A common solution is to derive a model of the system that is as close as possible to the real behavior of the plant, apply the input u_k to it and then use the estimated state $\hat{x}_k$. This estimate will also be unreliable due to the inevitable approximations of the model and to the process noise w_k. Here is where the state observer goes on stage: it merges the estimated state with the noisy measurement coming from the sensors to provide an optimal state estimate. The resulting distribution has a smaller variance and the mean value of the probability density function corresponds to the optimal estimation of the state [27].
Specifically, the Kalman Filter is able to converge to the real value of the states by means of a two-step prediction (notation$(^-)$ indicates estimate before measurement update and $(^+)$ indicates estimate after update):

A-Priori Estimate

The state and the error covariance at time k are predicted by the following equations

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \mathbf{A}_{k-1}\hat{\mathbf{x}}_{k-1}^+\mathbf{B}_{k-1}\mathbf{u}_{k-1} \\ \mathbf{P}_k^- &= \mathbf{A}_{k-1}\mathbf{P}_{k-1}^+\mathbf{A}_{k-1}^T + \mathbf{Q}_{k-1} \end{aligned} \tag{2.6}$$

A-Posteriori Estimate

On the basis of the measurements $\mathbf{y}_k$, the output of the prediction step is updated

$$\begin{aligned} \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k[\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-] \\ \mathbf{P}_k^+ &= [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^- \end{aligned} \tag{2.7}$$

where $\mathbf{K}_k$ is the optimal Kalman gain at time k, which have the purpose of weighting the effect of the information from the prediciton step and from the measurements according to the degree of reliability of each component as suggested by the variance of measurement and process noise. It is defined as follows

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T[\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k]^{-1} \tag{2.8}$$

$\mathbf{R}$ and $\mathbf{Q}$ are the error covariance matrices and $\mathbf{H}$ is the measurement matrix.

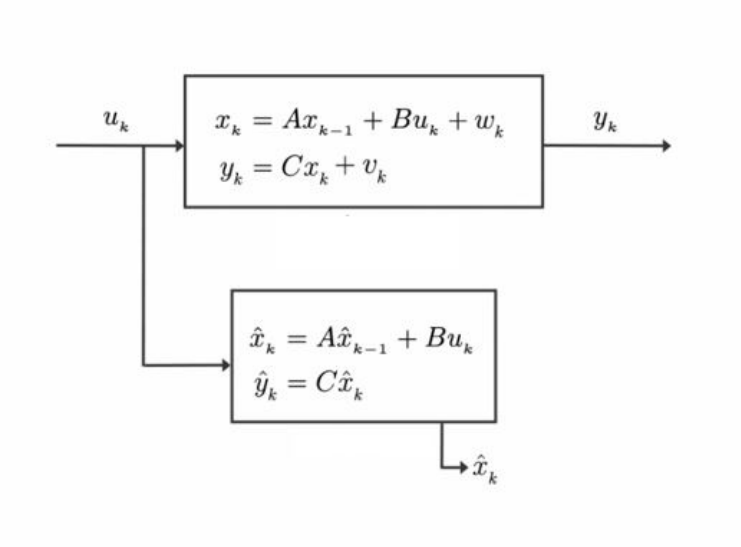


Figure 2.4: *Optimal state observer - graphical representation [27]*

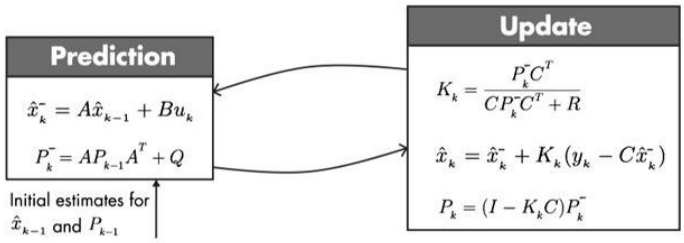


Figure 2.5: *Kalman Filter estimation steps [27]*

2.5 Non linear state estimators

The applications of the Kalman Filter are limited to **linear sytems**. What happens if we try to apply a KF to a nonlinear system?
Nonlinear equations that can describe a system model are 2.9 for continuous time and 2.10 for discrete time

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{w}(t), \\ \mathbf{y}(t) &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{2.9}$$

$$\begin{aligned} \hat{\mathbf{x}}_k &= \phi_{k-1}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_k, \\ \mathbf{y}_k &= h(\mathbf{x}_k) + \mathbf{v}_k \end{aligned} \tag{2.10}$$

One basic assumption of Kalman Filters is that the considered distribution must be gaussian. If the state transition function f(x) is a *linear* function, then the property is preserved and the probability density function maintains the zero-mean gaussian distribution after the application of the function.
If the transformation function is, instead, *nonlinear*, then the resulting distribution might not be gaussian and the filter might not be able to converge (the same holds for the measurement function h(x)).
The solution to this issue can be found in the design of **non linear state estimators**.

2.5.1 Extended Kalman Filter

One of the best known nonlinear state estimators is the Extended Kalman Filter. It solves the problem of estimating the states of nonlinear systems by linearizing the measurement and state transition functions around a point (the current state estimate) and applies Jacobians of the measurement and states to compute the covariance estimates using the same equations as in 2.6 and 2.7. Unlike the Kalman Filter, the EKF is not always optimal and stable, but it can be guaranteed to meet the expected performance only when the system meets specific requirements [5],[22],[29].
The equations becomes as follows

$$\begin{aligned} \hat{\mathbf{x}}_k^- &= \phi_{k-1}(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}), \\ \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k[\mathbf{y}_k + h_k(\hat{\mathbf{x}}_k^-)], \end{aligned} \tag{2.11}$$

and the Jacobians are defined in 2.12

$$\begin{aligned} \mathbf{A}_k &= \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}, \\ \mathbf{B}_k &= \frac{\partial f(\mathbf{x})}{\partial \mathbf{u}}, \\ \mathbf{H}_k &= \frac{\partial h(\mathbf{x})}{\partial \mathbf{x}} \end{aligned} \tag{2.12}$$

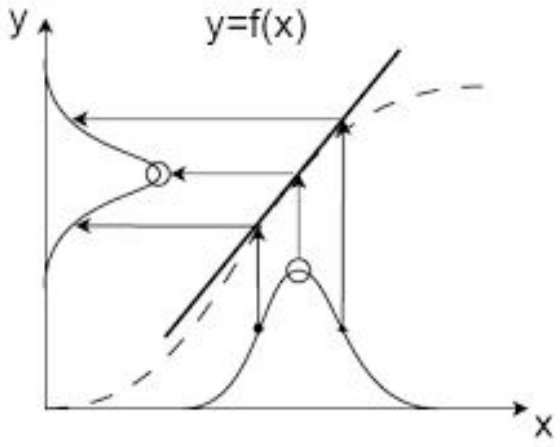


Figure 2.6: *Linearization performed by the Extended Kalman Filter*

2.6 Observability analysis

The concept of observability in control theory can be thought as a measure of how easily the states of the system can be retrieved knowing the inputs. The principle was first introduced by Rudolf E. Kálmán [18] for LTI systems and later extended to nonlinear ones by applying Lie derivatives [16].
From a control point of view, a system is said to be observable if, by considering two initial states $\mathbf{x}_1$,$\mathbf{x}_2 \in \mathbb{R}^n$, they are *distinguishible*, i.e. for every possible input $\mathbf{u}$, the corrisponding outputs $\mathbf{y}$ are not identical on their common interval of existance [16].
In other words, a system is said to be observable if the evolution of its states can be entirely determined by observing its outputs [3].
However, the property of observability of a system must not be considered as a static one that either applies to the case of study or it doesn't, but, as it will be shown in [2], it can be seen more as a local concept, that can be measured in terms of *degree* of observability for a given combination of states and inputs.
In the following, the methods for evaluating the observability of a system are described for linear time invariant and nonlinear systems.

2.6.1 Observability of LTI systems

Let's consider a LTI CT system Σ_l described by the following state-space form:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{2.13}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the *state vector*, $\mathbf{u}(t) \in \mathbb{R}^p$ is the *input vector*, $\mathbf{y}(t) \in \mathbb{R}^q$ is the *output vector*, while the matrices $A \in \mathbb{R}^{n,n}$, $B \in \mathbb{R}^{n,p}$, $C \in \mathbb{R}^{q,n}$ and $D \in \mathbb{R}^{q,p}$ are respectively the

state matrix, *input matrix*, *output matrix* and *feedforward matrix*.
A system of this form is said to be observable if the following matrix **O** is **full rank** [16]:

$$\mathbf{O} = \begin{bmatrix} C \\ CA \\ CA^2 \\ . \\ . \\ . \\ CA^{n-1} \end{bmatrix} \tag{2.14}$$

This condition is known as the *rank condition* and it is a valid preliminary index to determine whether, in general, the system is observable or not. The approach is valid also for non linear systems, where, however, the observability matrix has to be built using a different computational procedure.

2.6.2 Observability of nonlinear systems

Let's consider now a general continuous time nonlinear dynamic system Σ_{nl} described by the following model:

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}) \end{aligned} \tag{2.15}$$

where $\mathbf{x} = [x_1, x_2, ...x_n]^T \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^p$ is the input vector and $\mathbf{y} \in \mathbb{R}^m$ is the output vector.
In this case, the condition expressed for linear systems is not applicable anymore, since the condition 2.14 is a particular case of a more general condition [16].
For nonlinear systems, a definition of local weak observability can be given as follows ([3]). Assume that $\mathbf{x}_1$,$\mathbf{x}_2 \in \mathbb{R}^n$ are the initial states of 2.15 and V is an open set containing them. The two initial states are said to be V-distinguishable in time T at a local level if, for every input $\mathbf{u}$, the outputs are different for $t <= T$ for states that belong to V for $t <= T$.
The system Σ is **locally weakly observable** at $\mathbf{x}_0$ if there exists an open neighborhood U of $\mathbf{x}_0$ such that, for every open neighborhood V of x_0, the set of points V-indistinguishable from $\mathbf{x}_0$ coincides with $\mathbf{x}_0$. This condition must be verified for every $\mathbf{x} \in \mathbb{R}^n$. In other words, local weak observability at $\mathbf{x}_0$ corresponds to the possibility of finding at least one input $\mathbf{u}$ such that $\mathbf{x}_0$ is distinguishable from all its close states or a system is locally weakly observable if it is possible to istantaneously distinguish each initial state from its neighbors [3],[16],[31].
The former definition can guarantee that system Σ_{nl} is locally weakly observable at $\mathbf{x}_1$ for an input $\mathbf{u}$ if the following matrix **O** is full-rank for some index $k \in \mathbb{N}$:

$$\mathbf{O} = \begin{bmatrix} \nabla \mathcal{L}_f^0 h \\ \nabla \mathcal{L}_f^1 h \\ . \\ . \\ . \\ \nabla \mathcal{L}_f^k h \end{bmatrix} \tag{2.16}$$

The entries of **O** can be computed by recalling the definition of Lie derivatives of the measurement $\mathbf{y}$ [3],[16]:

$$\begin{aligned}
\mathcal{L}_f^0 h &= h(\mathbf{x}) \\
\mathcal{L}_f^1 h &= \nabla \mathcal{L}_f^0 h \cdot f \\
\mathcal{L}_f^2 h &= \nabla \mathcal{L}_f^1 h \cdot f \\
\mathcal{L}_f^3 h &= \nabla \mathcal{L}_f^2 h \cdot f \\
&\dots \\
\mathcal{L}_f^k h &= \nabla \mathcal{L}_f^{k-1} h \cdot f
\end{aligned} \tag{2.17}$$

2.6.3 Indices of observability

The observability of a system must not be considered as a property that is either owned or not, but rather as the *degree* of accuracy with which the estimation of the states can be conducted. In this regard, some measures to quantify the level of observability of a system have been introduced ([2]) which rely on the inspection of specific properties of matrix $\mathbf{O}$. In particular, the system results more observable if the matrix is *well conditioned*. The conditioning of the matrix cannot be reliably evaluated by observing only the determinant, as this observation does not give any information about how far the matrix is from being singular, but only tells us whether it is full rank or not. On the other hand, it can be shown ([11]) that a good way to perform this evaluation is to monitor its minimum singular value. A matrix with a very low minimum singular value will be ill-conditioned, while in the case where the minimum singular value is higher it will be better conditioned [11].

By keeping these definitions in mind, the *local estimation condition number* $\mathbf{C}$ of the nonlinear system has been defined as the ratio between the largest singular value and the smallest singular value of $\mathbf{O}$ ([2]). Favourable conditions on the system will result into a large $\mathbf{C}$, while a ill-conditioned estimation problem will give place to a rather small $\mathbf{C}$.
In this thesis, the inverse $\mathbf{C}^{-1}$ of the local estimation number has been adopted in place of $\mathbf{C}$. The choice is dictated by the greater handiness of $\mathbf{C}^{-1}$, which can have values in the range $[0, 1]$, leading to more readable plots.

$$C^{-1} = \frac{min(sing(\mathbf{O}))}{max(sing(\mathbf{O}))} \tag{2.18}$$

where $sing(\mathbf{O})$ are the local singular values of matrix $\mathbf{O}$ obtained through the singular values decomposition.

Chapter 3

Extended Kalman Filter for relative localization

The most important information for developing a flock of robots is knowing the location of all the agents in the 3D space. In order to make the swarm fully autonomous, it is necessary to allow each robot to reliably perform this estimation independently in their own body frame, without referring to an inertial frame common to the other robots. Therefore, the core of this thesis is to develop a localization algorithm able to estimate the 3D relative position of a MAVj (tracked) in the body frame of a MAVi (host), $\{j|j \in \mathbb{N}, j \neq i\}$ where $\mathbb{N} = \{1, 2, ..., N\}$, N number of robots in the flock within sight of MAVi.
The system is supposed to rely only on on-board equipment and to provide a localization in all the three dimensions. One important feature of the proposed solution is that the true height of the drones is assumed unknown and their relative vertical position is counted among the states to be estimated. In real-world applications an initial value for this parameter must be of course available, since the estimation concerns only the *relative* vertical spacing, but for these simulation that information is not needed.
In this chapter, the state observer designed to obtain the estimation is described. First, the available sensory equipment on the robotic platform is described in order to introduce the expected quality on the information in terms of noise on the measurements and to tailor the designed solution to the limitations of the gear. Then, the continuous time model of a system of two MAVs is described, followed by the discretized model of the same system and by the equations of the estimator. An Extended Kalman Filter has been chosen to address the problem for its computational power and low memory requirements, which make it suited for application on-board of small robots like the DelFly. Finally, a preliminary evaluation of the approach is proposed, which helps appreciating indicatively the feasibility of the solution.

3.1 Robotic platform: the DelFly Nimble

In order to provide an accurate description of the problem, it is important to understand the robotic platform considered and the available sensory inputs. In the following, a description of the DelFly Nimble is given with the purpose of providing an overview of the on-board equipment available and of the specific dynamics of the drone.

The most common robots used to implement flocking are wheeled robots, because they

make the problem easier by restricting the relative positioning to two dimensions and because they need to coordinate their motion only on the plane. Among the drones, the most fortunate are beyond any doubt quadrotors: there are some examples of application that use fixed-wing drones ([14]), but they make a less appealing solution due to their lower agility. Quadcopters, on the other hand, can virtually operate in 2D as their rest position is parallel to the ground and their yaw, pitch and roll angles have a limited amplitude. Moreover, they constitute an advantageous solution thanks to the relative simplicity of design, to the simple dynamics and to their ability to take-off vertically
The robot adopted in this work is, instead, a very unusual platform, since it does not belong to neither the fixed-wing family nor the quadrotors one: it is a flapper.
It may seem curious that the only flying solution that can be found in nature is that of a flapping wing, as it is adopted by all the species gifted with this enviable ability, and yet human solutions usually prefer fixed-wing and rotary-wing to emulate it, leaving projects like the ornithopter to dreaming geniuses like Leonardo da Vinci. Flappers in nature actually demonstrate a superior agility and quicker reactions compared to traditional drones, enabling complex maneuvres and a general greater nimbleness in navigating the environment. By imagining engineering applications of this flight strategy, it appears that a soft flapping wing would me more suited for employment near human beings, due to lower noise and a safer, lighter structure compared to a rotor, along with a more appealing impression due to the natural-looking shape of the robot.
For these reasons, reasearch to develop robotic platforms that mimic the flapping flight of animals has gained interest. Some examples of the state-of-the-art may be found in literature, like the Nano Hummingbird [28] and the Robobee [21]. The mentioned platforms, however, present some limitations in terms of possible maneuvres and flight autonomy. In this thesis, the robot model employed is that of a tiny, fully autonomous, free-flying robot inspired by fruit flies. The characterisctics of the DelFly Nimble are described in [20].

3.1.1 Flying mechanism and morphology

The robot is designed to emulate the behavior of flying insects. It is therefore tailless and consequently its motion in the space is obtained only through adjustmens of the orientation of the wings and of the flapping rate. It has two soft wings per side that can flap in counterphase, exploiting a clap and peeling mechanism [20]. Like many other flying robots and insects, it is able to control 4 degrees of freedom (DOFs) and to successfully achieve motion in a 6-DOFs space. The model allows to directly control roll, pitch and yaw orientation and the vertical flight through regulation of the thrust. The longitudinal motion is obtained with the body pitch, while the lateral one can be regulated by the roll angle, following the helicopter model.
Control over the 3D orientation is achieved by producing torques around the body axes: yaw torque through the wing root angle, roll torque by asymmetrical flapping frequency on the left and right side of the drone, pitch torque by changing the dihedral angle.
The main limitations of the platform are due to its small size, which determines a small payload and a low battery capacity, that allows an autonomy of about 5 minutes [20].
These drawbacks are compensated by the outstanding agility of the robot, which can perform angular accelerations around $5000°s^{-2}$ through 360° roll and pitch flips. The speed reaches 7 m/s in forward flight and 4 m/s in sideways. These characteristics result into a remarkably agile structure, able to perform sharp maneuvres and to move more freely in the 3D space.

Figure 3.1 shows the structure of the DelFly Nimble:

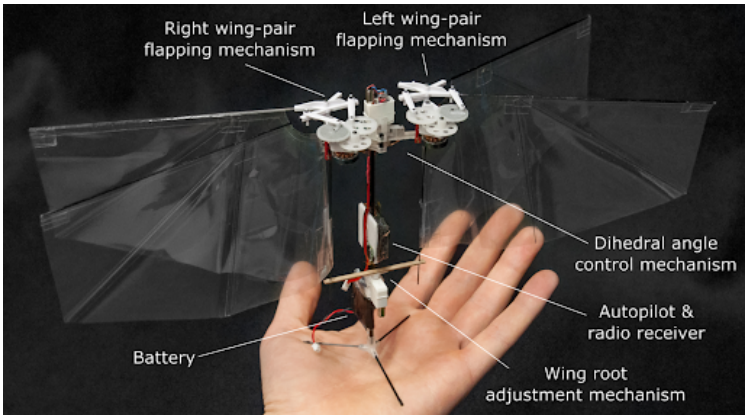


Figure 3.1: *DelFly Nimble ([20])*

3.1.2 Sensory equipment

The small payload makes it a must to reduce as much as possible the on-board equipment in order to extend the battery life and autonomy of the drone. The sensory hardware employed to develop this work is made up by sensors for attitude estimation, 3-axis gyroscope and accelerometer [20], a small autopilot and an optical flow sensor for measuring the velocities. An Ultra wide-band antenna is mounted on each robot to provide information on about the range between the host drone and any other robot within the working range of the UWB. The antenna is also exploited for communication among the drones, by broadcasting the information from the IMU and about the range through the flock. This solution has been selected because it is able to provide omnidirectional information, has a high communication rate and a low weight [15]. It seems also suited because it can satisfy both requirements of sensor and broadcasting device.

3.2 Kinematic model of a system of two MAVs in 3D

In this section the kinematic model of a two-robot system is introduced. First, the processing of the sensory inputs available to each drone is described in terms of common conventions; then, the derivation of the continuous time model of the swarm is addressed, along with a description of the results obtained by simulation of the system.

The sensory equipment described in 3.1 provides the necessary information to each drone for performing the relative positioning of any other robot within the UWB allowed range. For simplicity, the state observer designed for performing this estimation is applied to a generical system made up by two MAVs, where MAVi attempts to estimate the relative position of MAVj in its own body frame, $\{j|j \in \mathbb{N}, j \neq i\}$ where $\mathbb{N} = \{1, 2, ..., N\}$, N number of robots in the flock within sight of MAVi.
The available data coming from the sensors consists in the 3-axis velocity in the body frame $\mathbf{v}^b = [v^{b,x}, v^{b,y}, v^{b,z}]^T$ that can be obtained by fusing the optical flow measurements with the IMU ones; the yaw rate r^b, always in the body frame, coming from the gyroscope and

the range measurement d_{ij} provided by the ultra wide-band antenna, which measures the distance between MAVi and MAVj. Information about the true height is not used in this simulation, as the relative vertical position is numbered in the states to be estimated.
The fact that all the measurements performed by each drone are in its own body frame, a dissertation concerning the reference frames employed is called for. Since it is very difficult from the computational point of view to handle vectors in mobile body frames, but, at the same time, the setup of this thesis needs to avoid referring to a common inertial frame in order to spare the use of heading measurements, which, as explained in [15], are prone to interference and uncertainties due to disturbances in the local magnetic field as measured by magnetometers, a solution must be found to slim down calculations and maintain an autonomous approach to the problem. For all the models in the following analysis, an horizontal reference frame denoted by H_i, will be used, which is different from the body frame B_i fixed to each robot; frame H_i and frame B_i are both centered in the robot centre of mass (CoM), but, while B_i uses the three Euler angles to describe the drone's orientation in space, H_i corrects this orientation by the gravity vector $\mathbf{g}$, providing an horizontal frame parallel to the ground. For this reason, the quantities measured in the body-fixed frame B_i need to be transformed to H_i before being broadcasted by the UWB. It is possible to obtain the needed velocity and yaw rates by trasforming them using Cardan angles parametrization.

3.2.1 Euler angles

The common angles employed in the aerospace and maritime applications to describe the orientation of a body with respect of an inertial frame are the Euler angles, introduced in the 18^{th} century by Leonhard Euler. The Euler angles (RPY) can be described by the elementary rotations roll, pitch and yaw (ϕ, θ, ψ) around the mutually orthogonal basic vectors of the body frame. The order in which the roations are applied does matter: the rules of pre-multiplication of rotations around the fixed axis and of post.multiplication of rotations about the mobile axis are applied. This representation can be conveniently applied to decompose a proper orthonormal matrix $\mathbf{R}$ which describes the transformation between two frames that share the same origin, like the representation of the vectors in the body frame B_i into the horizontal frame H_i, which can be described as a multiplication of elementary rotations.

Elementary rotations

The rotations about the basic vectors of a Cartesian reference system are referred to as **elementary rotations** and are implemented by the following matrices:

$$\mathbf{R}(\mathbf{i}, \phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{bmatrix} \tag{3.1}$$

$$\mathbf{R}(\mathbf{j}, \theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) \\ 0 & 1 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{bmatrix} \tag{3.2}$$

$$\mathbf{R}(\mathbf{k},\psi) = \begin{bmatrix} cos(\psi_) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

The matrix decomposition can be performed as follows:

$$\begin{aligned} \mathbf{R} = \mathbf{R}(\phi,\theta,\psi) = \mathbf{R}(\mathbf{k},\psi)\mathbf{R}(\mathbf{j},\theta)\mathbf{R}(\mathbf{i},\phi) = \\ = \begin{bmatrix} c_\phi c_\psi - s_\phi c_\theta s_\psi & -c_\phi s_\psi - s_\phi c_\theta c_\psi & s_\phi s_\theta \\ s_\phi c_\psi + c_\phi c_\theta s_\psi & -s_\phi s_\psi + c_\phi c_\theta c_\psi & -c_\phi s_\theta \\ s_\theta s_\psi & s_\theta s_\psi & c_\theta \end{bmatrix} \end{aligned} \tag{3.4}$$

$$\begin{aligned} \mathbf{R} = \mathbf{R}(\phi,\theta,\psi) = \mathbf{R}(\mathbf{i},\phi)\mathbf{R}(\mathbf{j},\theta)\mathbf{R}(\mathbf{k},\psi) = \\ = \begin{bmatrix} c_\theta c_\psi & -c_\theta s_\psi & s_\theta \\ s_\phi s_\theta c_\psi + c_\phi s_\psi & -s_\phi s_\theta s_\psi + c_\phi c_\psi & -s_\phi c_\theta \\ -s_\theta c_\psi c_\phi + s_\phi s_\psi & c_\phi s_\theta s_\psi + s_\phi c_\psi & c_\phi c_\theta \end{bmatrix} \end{aligned} \tag{3.5}$$

where 3.4 indicates rotations about the mobile axis and 3.5 about fixed axis, while the notation $s(.)$ and $c(.)$ indicate respectively the $sin(.)$ and $cos(.)$.

3.2.2 Body frame and horizontal frame

As stated at the beginning of 3.2, the model of the system uses two different frames: the body-fixed frame B_i and the horizontal frame H_i. Both frames share the same origin, located at the CoM and therefore the transformation between them can be described by a simple composition of rotations. The peculiarity of frame H_i is that it is corrected by the gravity with respect to B_i, resulting into a reference frame with the z-direction othogonal to the ground pointing down and x-direction and y.direction forming a plane parallel to the ground. The two reference systems are shown in Figure **??**, where B_i is indicated with the label *body* while H_i appears as *inertial*.

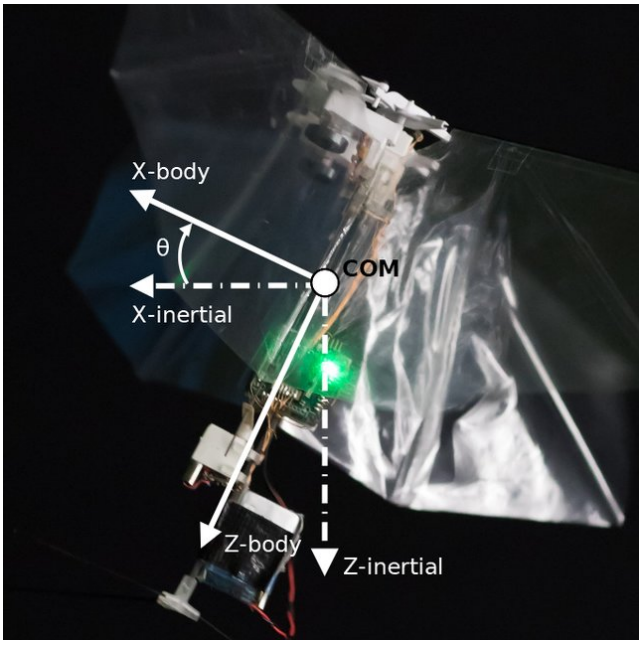


Figure 3.2: *Reference frames ([19])*

On the basis of the theoretical concepts described before, it is possible to ouline now the transformations before use that are needed to be applied to the quantities measured by on-board sensors. In particular, these involve the 3-axis body-frame velocity $\mathbf{v}^b$ and the yaw rate r^b, which have to be processed before being broadcasted to the rest of the flock.
It is possible to obtain the velocity in the horizontal frame $\mathbf{v}$ by rotating the vector in the body frame in X and Y, following this order:

$$\mathbf{v} = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ s(\phi)c(\theta) & c(\phi) & -c(\theta)s(\phi) \\ s(\theta)c(\phi) & s(\phi) & c(\phi)c(\theta) \end{bmatrix} \mathbf{v}^b \tag{3.6}$$

For what concerns the yaw rate r, it can be obtained by recalling the relationship between the angular velocity vector and the angular velocities measured in the body frame:

$$\omega = r^b \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + q^b \begin{bmatrix} -s(\psi) \\ c(\psi) \\ 0 \end{bmatrix} + p^b \begin{bmatrix} c(\psi)c(\theta) \\ s(\psi)c(\theta) \\ 0 \end{bmatrix} \tag{3.7}$$

where p^b indicates the roll rate measured by the gyroscope.

$$r = \frac{s(\psi)}{c(\theta)} q^b + \frac{c(\phi)}{c(\theta)} r^b \tag{3.8}$$

where q^b is the pitch rate as measured by the gyroscope.

3.2.3 Continuous time model

By considering two MAVs, it is possible to derive using the Newton formula the continuous time kinematic model $f(\mathbf{X}_{ij})$ describing the relative motion of the drones in the three dimensions. The model takes as input the velocities in the horizontal frame H_i and the yaw rates and computes the derivative of the relative position of MAVj in the transformed frame of MAVi. The input vector can be expressed as $\mathbf{U}_{ij} = [\mathbf{v}_i, \mathbf{v}_j, r_i, r_j]^T$ and the state vector as $\mathbf{X}_{ij} = [x_{ij}, y_{ij}, z_{ij}, \psi_{ij}]$. The prediction is obtained by considering the range between the two MAVs d_{ij}, which is the only observation available. Heights are not measured. The equations describing the model are the following, as in [26]:

$$\dot{\mathbf{X}}_{ij} = f(\mathbf{X}_{ij}) = \begin{bmatrix} R(\psi_{ij})\mathbf{v}_j - \mathbf{v}_i - Sr_i\mathbf{p}_{ij} \\ r_j - r_i \end{bmatrix} \tag{3.9}$$

where the 3-axis velocitites are expressed by $\mathbf{v}_j = [v_j^x, v_j^y, v_j^z]^T$ and $\mathbf{v}_i = [v_i^x, v_i^y, v_i^z]^T$, the yaw rates are r_j and r_i, $\mathbf{p}_{ij} = [x_{ij}, y_{ij}, z_{ij}]^T$ is the realtive position vector in the three dimensions, $\mathbf{R}(.)$ is the elementary rotation matrix about z that rotates j^{th} frame to i^{th} frame by the relative yaw angle, ψ_{ij}, ans $\mathbf{S}$ is the skew-symmetric matrix.

$$\mathbf{R}(\psi_{ij}) = \begin{bmatrix} cos(\psi_{ij}) & -sin(\psi_{ij}) & 0 \\ sin(\psi_{ij}) & cos(\psi_{ij}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

$$\mathbf{S} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

The relative state to be estimated is $\mathbf{X}_{ij} = [\mathbf{p}_{ij}, \psi_{ij}]$ and it has to be computed according to 3.10 by using the measurement of the range d_{ij} and the input vector $\mathbf{U}_{ij}$.

3.3 EKF for relative localization derivation

In this section the equations for implementing the Extended Kalman Filter for range-based relative localization are derived from the continuous time model of the system.
First, it is needed to discretize the equations 3.9, since the EKF works in DT (a CT would be also possible, but DT form is adopted in this thesis), obtaining the first equation in 3.12:

$$\begin{aligned} \hat{\mathbf{X}}_{k+1|k} &= F(\hat{\mathbf{X}}_k, \mathbf{U}_k) = \hat{\mathbf{X}}_k + \dot{\mathbf{X}}_k T_s, \\ \mathbf{P}_{k+1|k} &= \mathbf{A}_k \mathbf{P}_{k|k} \mathbf{A}_k^T + \mathbf{B}_k \mathbf{Q}_{k|k} \mathbf{B}_k^T \end{aligned} \tag{3.12}$$

where T_s is the time interval to update the EKF and it is assumed equal to the sampling time used in the simulation, $\hat{\mathbf{X}}_{k+1|k}$ is the vector of states predicted on the basis of $\hat{\mathbf{X}}_{k|k}$, the estimated state at the current time step. The second equation in 3.12 determines the prediction of the error covariance P, which is computed by considering the process noise covariance matrix Q (noise on the input) and the error covariance as predicted at the current time step $\mathbf{P}_{k|k}$. The matrices A_k and B_k denote respectively the state Jacobian matrix and the input Jacobian matrix. They can be derived as follows:

$$\mathbf{A} = \frac{\partial F}{\partial \mathbf{X}} = \begin{bmatrix} 1 & r_i T_s & 0 & T_s(-sin(\psi_{ij})v_j^x - cos(\psi_{ij})v_j^y) \\ -r_i T_s & 1 & 0 & T_s(cos(\psi_{ij})v_j^x - sin(\psi_{ij})v_j^y) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

$$\mathbf{B} = \frac{\partial F}{\partial \mathbf{U}} = \begin{bmatrix} -T_s & 0 & 0 & T_s cos(\psi_{ij}) & -T_s sin(\psi_{ij}) & 0 & T_s y_{ij} & 0 \\ 0 & -T_s & 0 & T_s sin(\psi_{ij}) & T_s cos(\psi_{ij}) & 0 & -T_s x_{ij} & 0 \\ 0 & 0 & -T_s & 0 & 0 & T_s & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -T_s & T_s \end{bmatrix} \tag{3.14}$$

The previous equations make up the prediction stage of the EKF. The outputs of this step are updated by merging this information with the measurement of the range, denoted by:

$$h = h(\mathbf{X}_{ij}) = \sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2} \tag{3.15}$$

from 3.15 it is possible to derive the equations for the update stage. In particular, the Jacobian matrix of the observation is obtained as:

$$\mathbf{H} = \frac{\partial h}{\partial \mathbf{X}} = [x_{ij}/h, y_{ij}/h, z_{ij}/h] \tag{3.16}$$

$$\begin{aligned} S &= (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}^T)^{-1} \\ y_{res} &= h(\hat{\mathbf{X}}_{k|k-1}) \\ \mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k^T S \\ \hat{\mathbf{X}}_k &= \hat{\mathbf{X}}_{k|k-1} + \mathbf{K}_k(z_k - y_{res}) \end{aligned} \tag{3.17}$$

Figure 3.3 shows the structure of the EKF as developed in Simulink. The Matlab function *EKF_predict* performs the prediction of the states $\mathbf{X}$ according to the equations 3.12; the ouputs of the stage are the predicted states $\hat{\mathbf{X}}_{k+1|k}$ and the predicted error covariance $\hat{\mathbf{P}}_{k+1|k}$ and they provide the input to the Matlab function *EKF_update*, where the formulas in 3.15 3.16 and 3.17 are implemented. The input u corresponds to the input vector $\mathbf{u}$ of velocities and yaw rates, to which a process noise with zero-mean white gaussian distribution v_k is added through the *Process noise block*. Input h is the measurement of the range as provided by the UWB, while the constants q_v and q_w are the variance of the error on the velocities and the yaw rates respectively.

3.4 Simulation with two drones

This section describes the preliminary simulation of the system performed with two drones in order to evaluate the performance of the filter. Realistic values have been assigned to the filter's parameter and to the amount of noise applied to measurement and inputs. The main indeces that are studied in order to quantify the efficiency of the estimator are the *convergence time* and the *quality of the prediction*.

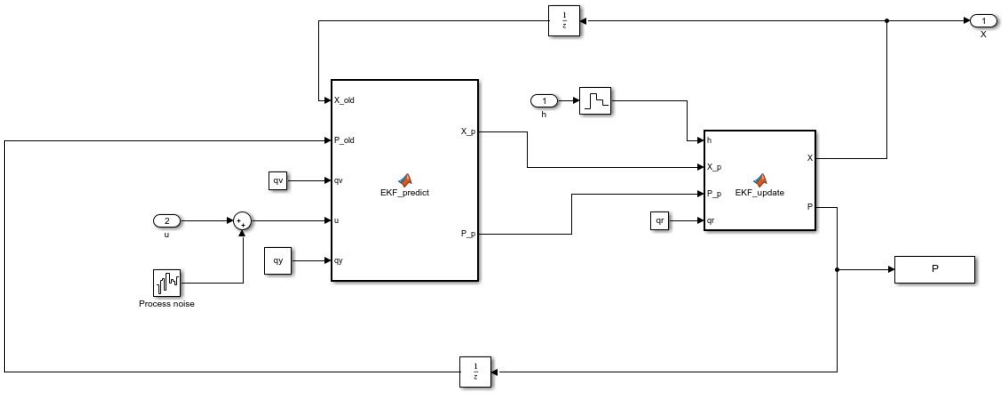


Figure 3.3: *Simulink model of the EKF structure*

The simulation has been run first using a simple simulator to check the behavior of the filter with a plant with simpler dynamics; afterwards, the model of the DelFly has been plugged in. The overall results are not significantly far away from each other, therefore, in the following, the reported experimental results are from the final simulation with the real drone model.

3.4.1 Parameters configuration and simulation environment

In this section the values for the parameters of EKF and simulation are reported.

Parameter	*Symbol*	*Value*
Sampling time	T_s	0.01 s
Input noise deviation - velocity	σ_v	0.25 m/s
Input noise deviation - yaw rate	σ_w	0.4 rad/s
Measurement noise deviation	σ_r	0.1 m

Table 3.1: *Simulation parameters*

The parameters for the EKF are computed as follows:

$$Q = \begin{bmatrix} \sigma_v^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_v^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_v^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_v^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_v^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_v^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_w^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \sigma_w^2 \end{bmatrix} =$$

$$= \begin{bmatrix} 0.25^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.25^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.25^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.25^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.25^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.25^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.4^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.4^2 \end{bmatrix}$$

$$R = \sigma_r^2 = 0.1^2 \tag{3.18}$$

Since 3.12 uses information from the previous time step, the EKF has to be initialized in terms of initial $\mathbf{P}$ and output vector $\mathbf{X}$. To avoid affecting the estimation in any way, the relative position of Robotj in Roboti frame is initialized at a random value, by assigning to x_j, y_j, z_j an arbitrary value in the range $[-5, 5]$ m. For what concerns the covariance matrix $\mathbf{P}$, it is assigned an initial value of :

$$\mathbf{P} = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

In 3.4 the Simulink model for the simulation is shown. The blocks "$Robot_j$" and "$Robot_i$" contain the DelFly Nimble model and the matlab function that performs the projection of the velocities from the body frame B_i to the horizontal frame H_i. The velocities in the horizontal frame and the yaw rates enter the subsystem named "Input", where they are merged to form the vector $\mathbf{u}$ that serves as input to the EKF. The positions in the inertial frame are, instead, employed to compute the range between the two drones in the block "Range computation": this compuatation is not actually performed by the drone, but it is needed to simulate the information provided by the UWB antenna. Finally, subsystem "Extended Kalman Filter" contains the filter and its structure is shown in 3.3.

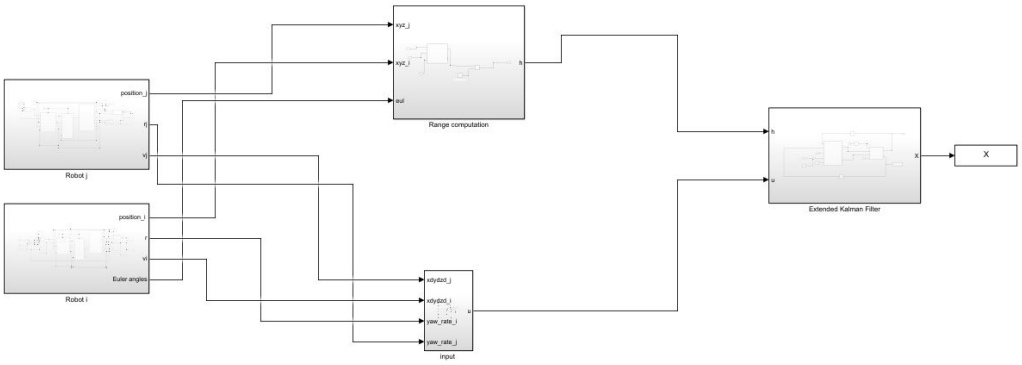


Figure 3.4: *Full simulink model*

The estimated states obtained through the state observer and simulation environment described in this chapter are shown in Figure 3.5. The red plot shows the real relative

position and relative yaw angle, while the blue plot refers to the same quantities as estimated by the EKF.
From this preliminary evaluation, it appears that the filter is capable of correctly converge to the true value, although this approximation takes longer than the 2D case, as it can be seen by comparing this results to those obtained by [26]. That model, which aimed at estimating only the relative position in the x and y direction, is able to converge in about 20s, while with the setup presented in this thesis the convergence time apears to be around 40s.
This delay was intuitively foreseen, as by adding the third dimension we are also increasing the ambiguity of the real state corresponding to the same set of inputs. This point will be discussed in detail in the following chapter dedicated to the observability analysis of the system.

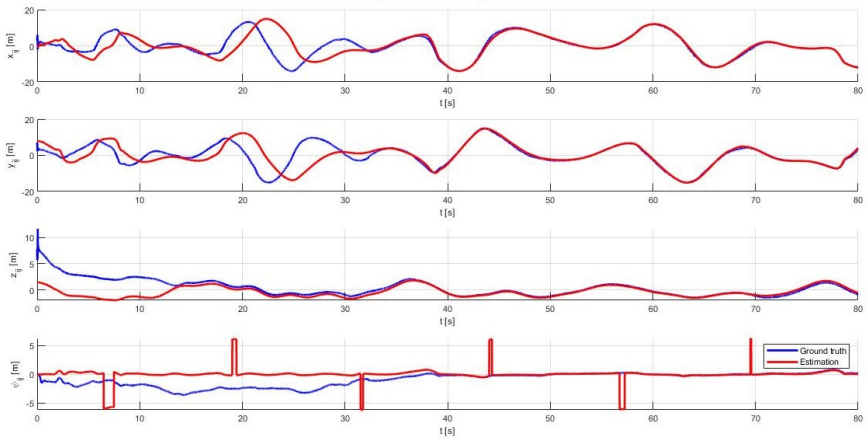


Figure 3.5: *Estimation of the four states through a range-based EKF*

In order to verify the correctness of convergence time, the system has been simulated multiple times for 80s under different initial conditions both in terms of the distribution of the robots and of initialization parameters of the filter. Figure 3.6, which shown the estimation error in the first three states, confirms that most of the runs are able to converge to a null estimation error around 40s.

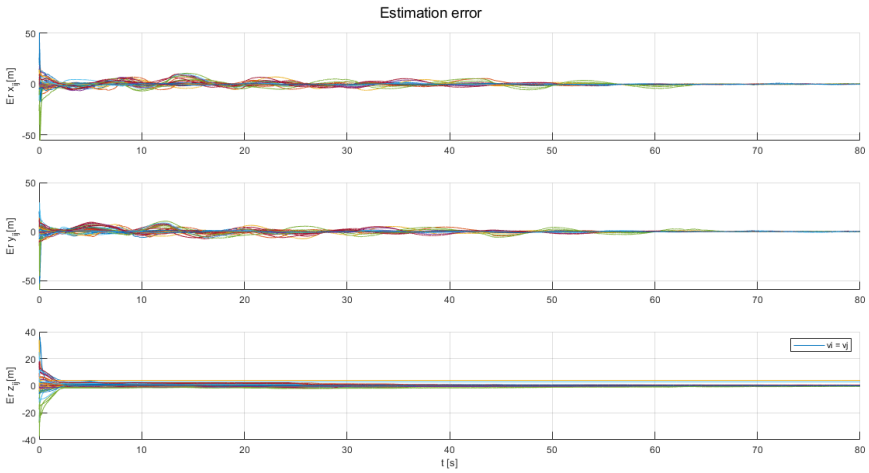


Figure 3.6: *Simulink model of the EKF structure*

Chapter 4

Observability analysis via Lie derivatives

This chapter is dedicated to the observability analysis of the system. The objective of the analysis is to identify possible combinations of inputs and states that could jeopardize the correct functioning of the filter.
The results obtained through this study are used to test the ability of the filter to converge to the true values of the states under observable conditions and in unobservable (or less observable) ones. The effort has proved useful in order to anticipate what to expect from the EKF and to design the second algorithm presented in Chapter 5, which relies heavily on the ability of each robot to correctly localize the preceding one in the flock.
As discussed throughly in Chapter 2, a common method for studying the observability of nonlinear systems, like the one considered in this work, is to build an observability matrix **O** through Lie derivatives and to analyze its properties in terms of rank and determinant.

In the following, first the mathematical procedure to extract the entrances of matrix **O** from the system equations and to compute its determinant is carried out. Once the results of the operation are derived, they are analyzed to deliver a preliminary evaluation of the properties of the matrix by checking the rank condition and evident unobservable conditions. At this point, the degree of observability of the system is studied by introducing a measure of observability, the *local estimation condition number* ([2] and [3]). The purpose of this step is to give a qualitative overview on how the observability of the system is affected by variations of the inputs and of the states and to perform a comparison with the 2D case ([26] and [15]). Finally, the theoretical results are verified by experimental evidence obtained through simulation by testing the EKF performance with different trajectories imposed to the drones.

4.1 Observability matrix derivation

In this section, the entrances of the observability matrix **O** are computed from the system equations using Lie derivatives in order to perform a local weak observability analysis of the EKF for range-based relative localization in 3D are computed.

Let's consider the non-linear system Σ describing the problem under study in Chapter 3 through the following state-space equations:

$$\begin{aligned} \dot{\mathbf{X}}_{ij} &= f(\mathbf{X}_{ij}) = \begin{bmatrix} R(\psi_{ij})\mathbf{v}_j - \mathbf{v}_i - Sr_i\mathbf{p}_{ij} \\ r_j - r_i \end{bmatrix} \\ \mathbf{y} &= h(\mathbf{X}_{ij}) = d_{ij} \end{aligned} \tag{4.1}$$

where $\mathbf{X}_{ij} = [x_{ij}, y_{ij}, z_{ij}, \psi_{ij}]^T$ is the state vector, $\mathbf{u} = [\mathbf{v}_i, \mathbf{v}_j, r_i, r_j]$ is the input vector and $\mathbf{y}$ is the measurement of the range. The states indicate the relative position of MAVj with respect to MAVi $\mathbf{p} = [x_{ij}, y_{ij}, z_{ij}]^T$ and the relative yaw angle ψ_{ij}, the inputs are the 3-axis velocities and the yaw rates of the two MAVs in the horizontal frame H_i. $d_{ij} = \sqrt{x_{ij}^2 + y_{ij}^2 + z_{ij}^2}$ is the measured range.
By observing the structure of the problem, it is possible to notice that the number of states to be estimated is four. Therefore, in order to study the local weak observability of the system according to the theory expressed in Chapter 2, the derivation of Lie derivatives must reach the third order. The rank condition states, in fact, that for a system to be *locally weakly observable* the observability matrix must be full rank [16]. For this reason, matrix $\mathbf{O}$ will have the following shape:

$$\begin{bmatrix} \nabla\mathcal{L}_f^0(\mathbf{h}) \\ \nabla\mathcal{L}_f^1(\mathbf{h}) \\ \nabla\mathcal{L}_f^2(\mathbf{h}) \\ \nabla\mathcal{L}_f^3(\mathbf{h}) \end{bmatrix} = \begin{bmatrix} \partial\mathcal{L}_f^0 h/\partial\mathbf{X}_{ij} \\ \partial\mathcal{L}_f^1 h/\partial\mathbf{X}_{ij} \\ \partial\mathcal{L}_f^2 h/\partial\mathbf{X}_{ij} \\ \partial\mathcal{L}_f^3 h/\partial\mathbf{X}_{ij} \end{bmatrix} \tag{4.2}$$

In order to simplify the equations, the observation can be expressed in the power format $\frac{1}{2}\mathbf{p}^T\mathbf{p}$ for the following computations.
In this section, the notations $\mathbf{p}_{ij}$, $\mathbf{R}$, $\mathbf{S}$, $\mathbf{v}_i$, $\mathbf{v}_j$ are referred to the 2D components of the quantities, so they stand for the x and y components of the vectors, while the z component is written separately. The choice has the double purpose of making the formulas more readable and of highlighting the effect due to the addition of the third dimension: as it will be discussed in the following, the behavior of the system along the vertical direction is often detached from the one in the plane and the two affect the observability of the system separately.
The derivation of the Lie derivatives is then performed as follows:

$$
\begin{aligned}
\mathcal{L}_f^0 h &= h(\mathbf{X_{ij}}) = \frac{1}{2}\mathbf{p}^T\mathbf{p} \\
\nabla\mathcal{L}_f^0 h &= [x_{ij},\ y_{ij},\ z_{ij},\ 0] \\
\mathcal{L}_f^1 h &= \nabla\mathcal{L}_f^0 h \cdot f \\
\nabla\mathcal{L}_f^1 h &= [(R\mathbf{v}_j - \mathbf{v}_i)^T,\ v_j^z - v_i^z,\ \mathbf{p}_{ij}^T RS\mathbf{v}_j] \\
\mathcal{L}_f^2 h &= \nabla\mathcal{L}_f^1 h \cdot f \\
\nabla\mathcal{L}_f^2 h &= [(-r_i S\mathbf{v}_i + r_j RS\mathbf{v}_j)^T,\ 0,\ -2\mathbf{v}_i RS\mathbf{v}_j - r_j\mathbf{p}_{ij}^T R\mathbf{v}_j] \\
\mathcal{L}_f^3 h &= \nabla\mathcal{L}_f^2 h \cdot f \\
\nabla\mathcal{L}_f^3 h &= [(r_i^2\mathbf{v}_i - r_j^2 R\mathbf{v}_j)^T,\ 0,\ 3(r_j - r_i)\mathbf{v}_i^T R\mathbf{v}_j - r_j^2\mathbf{p}_{ij}^T RS\mathbf{v}_j]
\end{aligned}
\tag{4.3}
$$

The equations in 4.3 are used to build by substitution the observability matrix $\mathbf{O}$:

$$
\mathbf{O} = \begin{bmatrix}
\mathbf{p}_{ij} & z_{ij} & 0 \\
(R\mathbf{v}_j - \mathbf{v}_i)^T & v_j^z - v_i^z & \mathbf{p}_{ij}^T RS\mathbf{v}_j \\
(-r_i S\mathbf{v}_i + r_j RS\mathbf{v}_j)^T & 0 & -2\mathbf{v}_i^T RS\mathbf{v}_j - r_j\mathbf{p}_{ij}^T R\mathbf{v}_j \\
(r_i^2\mathbf{v}_i - r_j^2 R\mathbf{v}_j)^T & 0 & 3(r_j - r_i)\mathbf{v}_i^T R\mathbf{v}_j - r_j^2\mathbf{p}_{ij}^T RS\mathbf{v}_j
\end{bmatrix}
\tag{4.4}
$$

The study that will follow in the upcoming sections will require the comparison with the observability in the 2D system. To this end, the observability matrix obtain in this case in [26] is reported here and indicated as $\mathbf{O}^{2D}$.

$$
\mathbf{O}^{2D} = \begin{bmatrix}
\mathbf{p}_{ij} & 0 \\
(R\mathbf{v}_j - \mathbf{v}_i)^T & \mathbf{p}_{ij}^T RS\mathbf{v}_j \\
(-r_i S\mathbf{v}_i + r_j RS\mathbf{v}_j)^T & -2\mathbf{v}_i^T RS\mathbf{v}_j - r_j\mathbf{p}_{ij}^T R\mathbf{v}_j
\end{bmatrix}
\tag{4.5}
$$

The properties of the matrix in 4.4 are used to study the local weak observability of the system under different conditions in the following section.

4.2 Rank condition verification

The first verification that is done is to check that the matrix is full rank, meaning that its determinant is not null. This is called the rank condition and ensures that the matrix is well-posed. It is a verification that can tell whether, in general, given these inputs and these measurements, the set of states can be estimated by observing the outputs. The ranl condition can be immediately verified with MATLAB.
Since this method is very general, it is important to determine the conditions that can lead to the loss of this property.

4.2.1 Intuitively unobservable conditions

After obtaining the shape of the observability matrix 4.4, it is possible to make some preliminary observations concerning its rank.
As it can be noticed, the third column depends only on the relative motion and position of the drones in the vertical direction. One intuitively unobservable condition is therefore found in the case where the robots belong to the **same horizontal plane** and are **not moving vertically** or are moving vertically **at the same velocity**: if the relative velocity in z goes to zero and the drones happen to be on the same horizontal plane ($z_{ij} = 0$), then the third column becomes null and the matrix loses rank.
Another unobservable condition would occur if the two MAVs occupied the same spot in space: this is not possible due to their physical dimension, that prevents them from overlapping. This result is, however, interesting if compared to the study in 2D carried out in [15] and [26]: in that case, the EKF estimated the relative position of MAVj with respect to MAVi in x and y directions and from the observability analysis it resulted that if the two MAVs where to find themselves in the same position in x and y, but separated by height, the system would have been unobservable. Here, on the other hand, the same situation appears not to lead to unobservability, since, as it appears from 4.4, the decoupled nature of the behavior in z from the behavior on the horizontal plane, gurantees that the observability matrix preserves its rank as long as at least one among x_{ij}, y_{ij}, z_{ij} is non-zero.
A third intuitive condition that would lead to unobservability of the system can be drawn by observing that all the terms but the first row depend on the velocity of MAVj, $\mathbf{v}_j$. It appears that MAVi is able to correctly estimate the relative position of MAVj only if the latter is moving. The opposite is not true, as even in the case where MAVi is stationary, it is still capable of correctly carrying out the estimation.
The intuitive unobservable conditions are summarized as follows:

1. $z_{ij} = 0$ and $v_j^z = v_i^z$
2. $\mathbf{p} = \mathbf{0}$
3. $\mathbf{v}_j = 0$

Figure 4.1 shows a graphical representation of how unobservable conditions found in [15] and [26], become only limit condition with the 3D EKF:

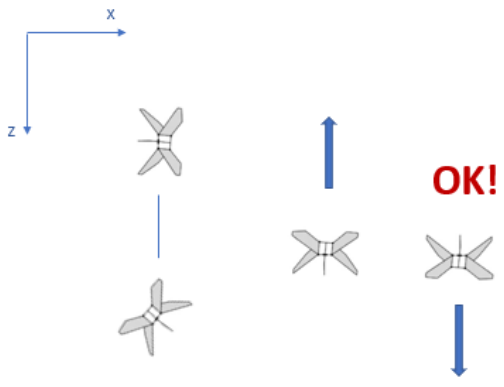


Figure 4.1: *Limit conditions to the observability of the system*

If the drones lie on the same vertical direction the system is still observable; if they belong to the same horizontal plane and their relative velocity in such plane is the same, they can localize each other provided that the relative vertical velocity is different from zero. Thi findings suggest that the number of situations in which observability is guaranteed are increased with respect to the application in 2D.

4.3 Determinant computation

The previous conditions can be verified also by computing the determinant of **O**. Now, the task appears really hard given the complexity of the matrix.
The procedure followed to extract $|\mathbf{O}|$ is to replace all the huge terms with letters and procede with a symbolic cofactor expansion in cascade along the third column, order the result conveniently and then replace the letters with the original entries of the matrix. All of the computations have been double-checked with the MATLAB Symbolic Math Toolbox.

4.3.1 Mathematical procedure

First, the larger terms are replaced in a symbolic manner:

$$\mathbf{O} = \begin{bmatrix} x_{ij} & y_{ij} & z_{ij} & 0 \\ a & b & v_j^z - v_i^z & A \\ c & d & 0 & B \\ e & f & 0 & C \\ \end{bmatrix} \tag{4.6}$$

Now, the determinant is derived through cofactor expansion. The two cofactors can be computed according to the following formulas:

$$\mathbf{M}_1 = (-1)^4 \, z_{ij} \, det \begin{bmatrix} a & b & A \\ c & d & B \\ e & f & C \end{bmatrix} \tag{4.7}$$

and

$$\mathbf{M}_2 = (-1)^5 \, (v_j^z - v_i^z) \, det \begin{bmatrix} x_{ij} & y_{ij} & 0 \\ c & d & B \\ e & f & C \end{bmatrix} \tag{4.8}$$

Given the complexity of each term, they can be further expanded as the sum of the determinant of two cofactors. In particular, cofactor expansion is performed for both $\mathbf{M}_1$

and $\mathbf{M}_2$ along the last column. The resulting expressions are in 4.10

$$\begin{aligned}\mathbf{M}_1 &= z_{ij}\,[A(cf-de)-B(af-be)+C(ad-bc)]\\ \mathbf{M}_2 &= -(v_j^z-v_i^z)\,[-B(fx_{ij}-ey_{ij})+C(dx_{ij}-cy_{ij})]\end{aligned} \tag{4.9}$$

The determinant can therefore be computed as the sum of the two:

$$det(\mathbf{O}) = \mathbf{M}_1 + \mathbf{M}_2 \tag{4.10}$$

4.3.2 Terms ordering and analysis

The form of 4.9 and 4.10 is not convenient for verifying the intuitive conditions previously observed, nor for identifying additional unobservable situations. In the following, the terms are manipulated and written in a handy manner.

Numerical expression of $\mathbf{M_2}$

We first consider $\mathbf{M}_2$ and we notice that it can be seen as made up by two parts with $-(v_j^z - v_i^z)$ as a common coefficient, both depending on the relative position of j with respect to i on the horizontal plane ($\mathbf{p}_{ij}$): we can therefore try to write $\mathbf{M}_2$ as a function of $\mathbf{p}_{ij}$.

$$\mathbf{M}_2 = -(v_j^z - v_i^z)[-B(fx_{ij} - ey_{ij}) + C(dx_{ij} - cy_{ij})]$$

Coefficient $-(v_j^z - v_i^z)$

TERM 1 $-B(fx_{ij} - ey_{ij})$

$$-B(fx_{ij} - ey_{ij}) = -B[e\ f]\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}x_{ij}\\ y_{ij}\end{bmatrix} = -B[e\ f]S\mathbf{p}_{ij} \tag{4.11}$$

TERM 2 $C(dx_{ij} - cy_{ij})$

$$C(dx_{ij} - cy_{ij}) = C[c\ d]\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}x_{ij}\\ y_{ij}\end{bmatrix} = C[c\ d]S\mathbf{p}_{ij} \tag{4.12}$$

The numerical form of M_2 can be written as follows:

$$\begin{aligned}\mathbf{M}_2 = -(v_j^z - v_i^z)[&(2\mathbf{v}_i^T RS\mathbf{v}_j + r_j\mathbf{p}_{ij}^T R\mathbf{v}_j)(r_i^2\mathbf{v}_i - r_j^2 R\mathbf{v}_j)^T\\ &+ (3(r_j - r_i)\mathbf{v}_i^T R\mathbf{v}_j - r_j^2\mathbf{p}_{ij}^T RS\mathbf{v}_j)(-riS\mathbf{v}_i + r_j RS\mathbf{v}_j)^T]S\mathbf{p}_{ij}\end{aligned} \tag{4.13}$$

What to notice about the previous result:

- All of the terms depend on v_j
- All of the terms depend on $\mathbf{p}_{ij}$
- All of the terms depend on $Rv_j^z - v_i^z$
- All of terms depend on a factor that goes to zero if the velocities in the plane are parallel

Numerical expression of $\mathbf{M_1}$

Now the first half of the determinant, $\mathbf{M}_1$, is to be considered.

$$\mathbf{M}_1 = z_{ij}[A(cf - de) - B(af - be) + C(ad - bc)]$$

Coefficient z_{ij}

First it can be noticed that the second and the third terms depend on a and b, so it is possible to consider those two terms separately from the first term $A(cd - de)$ in order to arrange them to collect a and b.

TERM 2 $-B(af - be)$

$$-B[e\ f]\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}a\\ b\end{bmatrix} = -B[e\ f]S[a\ b]^T \tag{4.14}$$

TERM 3 $C(ad - bc)$

$$C[c\ d]\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}a\\ b\end{bmatrix} = C[c\ d]S[a\ b]^T \tag{4.15}$$

In this way, **TERM 2** and **TERM 3** can be rewritten as a function of $[ab]^T$ ain the following way: $(C[c\ d] - B[e\ f])S[a\ b]^T$.
It is noteworthy that $[a\ b] = (R\mathbf{v}_j - \mathbf{v}_i)^T$, which actually express the velocity difference in the ith frame. This means that this part of the determinant becomes zero for parallel velocities of the two robots in the horizontal plane.
At this point, the first term of M_1 is studied.

TERM 1 $A(cf - de)$

It can be noticed that $A = \mathbf{p}_{ij}^T RS\mathbf{v}_j = \mathbf{v}_j^T RS\mathbf{p}_{ij}$, which actually depends on the same term $S\mathbf{p}_{ij}$ as the expanded $\mathbf{M}_2$.
By manipulating the term as done with M_2, it is possible to obtain:

$$A(cf - de) = A[e\ f]S[c\ d]^T \tag{4.16}$$

and by substituing the terms:

$$A(cf - de) = (r_i^2\mathbf{v}_i - r_j^2 R\mathbf{v}_j)^T S(r_i S^T\mathbf{v}_i - r_j RS^T\mathbf{v}_j)v_j^T RS\mathbf{p}_{ij} \tag{4.17}$$

With the information about the expanded terms of $\mathbf{M}_1$, it is possible to write this part of the determinant as follows:

$$\begin{aligned}\mathbf{M}_1 = z_{ij}\{&(r_i^2\mathbf{v}_i - r_j^2 R\mathbf{v}_j)^T S(r_i S^T\mathbf{v}_i - r_j RS^T\mathbf{v}_j)\mathbf{v}_j^T RS\mathbf{p}_{ij} +\\ &[(3(r_j - r_i)\mathbf{v}_i^T R\mathbf{v}_j - r_j^2\mathbf{p}_{ij}^T RS\mathbf{v}_j)(-r_i S\mathbf{v}_i + r_j SR\mathbf{v}_j)^T +\\ &(2\mathbf{v}_i^T RS\mathbf{v}_j + r_j\mathbf{p}_{ij}^T R\mathbf{v}_j)(r_i^2\mathbf{v}_i - r_j^2 R\mathbf{v}_j)^T]S(R\mathbf{v}_j - \mathbf{v}_i)\}\end{aligned} \tag{4.18}$$

What to notice about the previous result:

- All of the terms depend proportionally on $Rv_j - v_i$
- All of the terms depend on z_{ij}

It is interesting to notice that $\mathbf{M}_1$ depends on the relative position in the vertical direction, but not on the relative position in the horizontal direction. On the other hand, $\mathbf{M}_2$ depends on the relative position $\mathbf{p}_{ij}$ in the plane, but not on the relative height. These two terms suggest that the system can be observed as long as

- the drones belong to the same plane, but occupy different (x,y) positions
- the drones occupy the same (x,y) position, but they belong to different planes
- the drones occupy different (x,y,z) positions

For what concerns the velocities, since all of the terms depend on the velocity difference in the x and y direction $\mathbf{v}_i - R\mathbf{v}_j$ and on $\mathbf{v}_j$ alone, two general conditions to guarantee observability are that the "tracked" robot must be moving in the plane and that the velocities in the plane must be non parallel.
The velocity difference in the z direction affects only one term ($\mathbf{M}_2$), therefore intuitively having the same vertical direction might make the system *less* observable, but not unobservable.

4.4 Results from the observability matrix O determinant analysis

Since this system is the 3D version of the one studied in 2D by [15], it is possible to compare the two results.
In the 2D case, the conditions for unobservability were the same found here for what concerns the velocities (i.e. $\mathbf{v}_j = 0$ or $\mathbf{v}_i = R\mathbf{v}_j$) and $\mathbf{p}_{ij} = 0$.
On the other hand, the 3D system seems to guarantee observability in a larger number of cases. In fact, in the 3D case the condition linked to the relative position in the z component ensures observability where in the 2D case the system would have turned out unobservable, specifically the system remains observable even when the (x,y) coordinates coincide, provided that the planes are different. In the 2D system the relative vertical position was not considered, so the coincidence in (x,y) was enough to lose observability. This is due to the fact that z_{ij} and $\mathbf{p}_{ij}$ do not appear in all the terms of the determinant concurrently and therefore the presence of one of the two conditions does not send the entire determinant to 0.
To wrap it up, the following unobservable situations have been found:

- the two robots occupy the same spot in space
- the two robots move with parallel velocities in the three directions
- the two robots travel with parallel velocities in the z direction *and* they belong to the same horizontal plane
- the tracked robot (j) is still

The occurrance of partial conditions (like $z_{ij} = 0$ or $\mathbf{p}_{ij} = 0$ for example) might intuitively make the system less observable, because the value of $|\mathbf{O}|$ becomes smaller, but the fact that there is no term in the determinant expression that depend concurrently on both parameter suggests that a weak observability of the system is maintained.

This analysis has been useful to understand the limit conditions that guarantee the correct working of the relative position estimator and to identify the situations that lead to unobservability the 2D system, but not the 3D one. It is not possible, however, to appreciate from the rank condition if the observability of the system is enough to guarantee the correct functioning of the estimation.
It would be interesting to study *how well* conditioned the system is, or, in other words, how easy it is to distinguish uniquely one state from the other basing on the measurement $\mathbf{h}$.
Some interesting parameters have been introduced in [2], which are based on the observability gramian and other indeces that can be estracted from it.
In the present study, the inverse of the *local estimation condition number* will be used.

4.5 Measures of observability

The rank verification can be a valid method for understanding whether a system is locally weakly observable or not, but it does not provide any information about the degree of observability, i.e. how observable the system is. In this section we study the effect of the inputs and of the states combined on the level of observability of the model by monitoring the changes that occur in the value of the inverse of the *local estimation condition number* C^{-1} presented in Chapter 2. This parameter helps us to understand how well posed the matrix $\mathbf{O}$ is, or, in other terms how far it is from being singular. High values of C^{-1} indicate a better conditioning, while low values suggest that the current disposition of the agents does not allow a good estimation of the states.
The analysis has been carried out by showing graphically the variation of the local estimation condition number in the colormaps and by building level maps. Different parameters of the system are checked by fixing all the other terms and varying only the one of interest around a given initial set of conditions. The results are compared to those obtained in the 2D case in order to see how affected the observability of the system is by the estimation of the relative position in the third dimension. Finally, some relevant cases are identified and studied in detail through simulation of the system with these particular conditions.

4.5.1 Unobservable conditions

The first verification that is carried out concerns the unobservable conditions detected in the previous section. This evaluation is useful not only to validate the analytical findings, but also to confirm the correctness of the method that we undertake to implement.
In Figure 4.2, the three unobservable cases highlighted in Section 4.2.1 are shown: the index C^{-1} is studied by fixing all the parameters but the relative position in the horizontal plane of the two robots, p_x and p_y, which vary around the initial position $p_x = 0$ m and $p_y = 0$ m in the range $[-5, 5]$ m. The colorbar beside each plot indicates that $C^{-1} = 0$ for any relative position of the robots for all the three cases: parallel velocities in the three directions $\mathbf{v}_i = kR\mathbf{v}_i$, MAVj still $\mathbf{v}_j = 0$ m/s and drones on the same horizontal plane $z_{ij} = 0$ m with the same vertical velocity $v_j^z = v_i^z$.

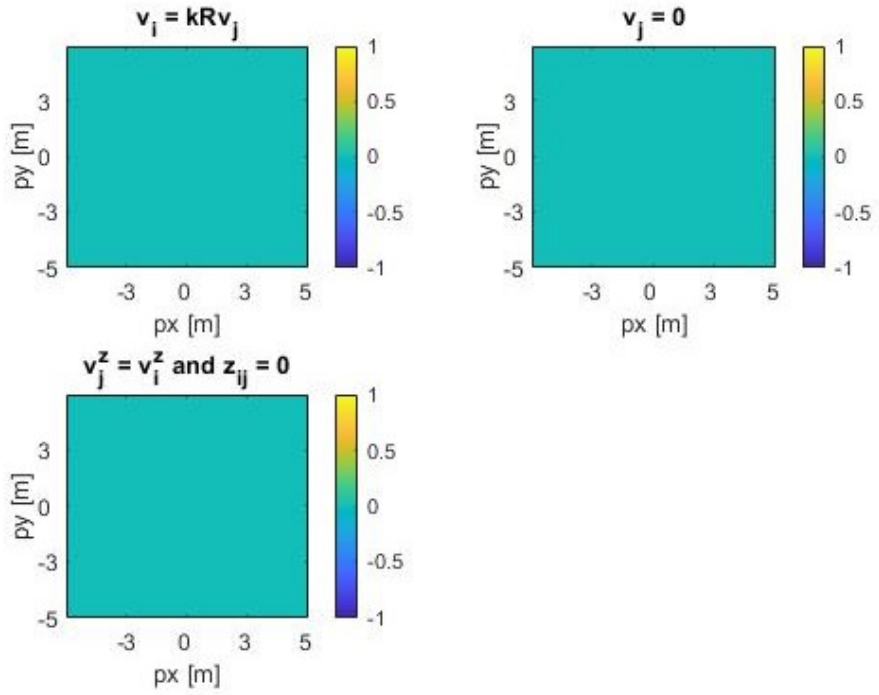


Figure 4.2: *Sensitivity to measurement noise*

4.5.2 Comparison with the 2D case

In this section the effect of all the relevant parameters on the observability index is studied in the 2D and 3D case. Since the observability of the system is defined as the possibility of obtaining distinguishable outputs as a result of the application of different inputs [16], intuitively the expected result is to find that the new filter results less observable compared to the 2D case; in fact, now the ambiguity in the determination of the true relative position of MAVj is increased due to the fact that the same measured range associated to certain velocity vectors can correspond to an agent belonging to a sphere of radius d_{ij} around MAVi, while in 2D the ambiguity is only between two possible positions of MAVj with respect to MAVi on the plane (left or right). The results of the study confirm this supposition and are reported in the following.

Before building the colormaps it is necessary to study the effect of two parameters on the overall behavior of the system, as they can affect differently the 2D and 3D filter. It is, in fact, important to provide the same conditions to both the systems under study, in order to avoid setting up a combination of parameters that can be favourable to one of the two, but not to the other. To this end, the effect of the relative yaw angle ψ_{ij} and of the relative vertical position z_{ij} on the observability of the systems is plotted.
For the simulation to estimate the effect of ψ_{ij} and of z_{ij}, the parameters of the system are fixed as shoen in 4.1.

Parameter	Value
$\mathbf{p}_{ij}$	$[2,2]^T$ m/s
z_{ij}	0 m
$\mathbf{v}_i$	$[-2,-2,-2]^T$ m/s
$\mathbf{v}_j$	$[5,5,5]^T$ m/s
ψ_{ij}	160°
r_i	$10°/s$
r_j	$-20°/s$

Table 4.1: *Parameters for the analysis of ψ_{ij} and z_{ij}*

The relative height z_{ij} is set to 0 m in order to test the two systems under the same conditions.

Effect of ψ_{ij} on the observability index

The first parameter to be studied and compared is the relative yaw angle ψ_{ij}. The importance of this preliminary evaluation is dictated by the need of fixing this value in all the other simulations where the focus is on the impact of other inputs on C^{-1}: the choice of ψ_{ij} appears therefore not trivial, as it will set one common condition to the two systems.
In Figure 4.3 the relative yaw angle varies between 0° and 180° in order to see the trend of C^{-1} for all the possible relative orientations of the two MAVs.

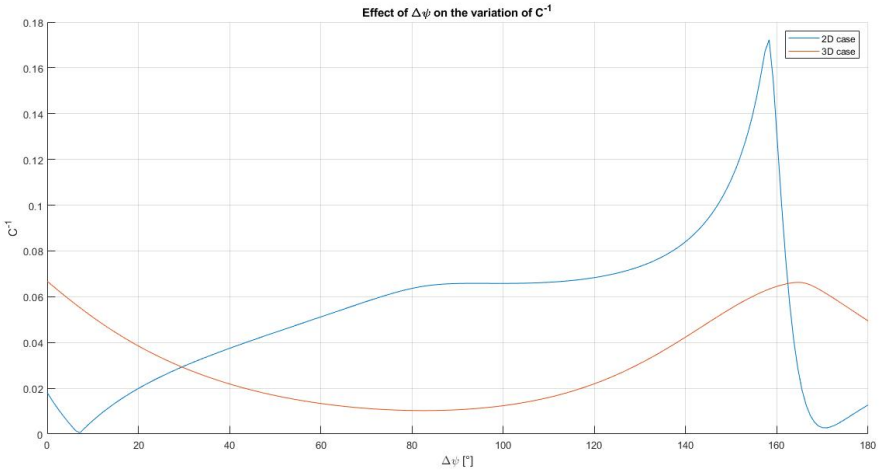


Figure 4.3: *Effect of relative vertical position*

The blue plot corresponds to the behavior oserved in the 2D: as expected by considering the results in [15], the observability of the system drops for values close to 0° and 180°, which correspond to parallel velocities, while it shows a positive tendency between 10° and 160° with a significant peak around this last value. On the overall, the observability ondex takes values between 0 when the MAVs are parallel and 0.17 for the best conditioned case.

An interesting result is found, instead, for the 3D case. The first observation that can be made is that the unobservable condition ($C^{-1} = 0$) is never reached, thanks to the vertical dynamics imposed to the system, which prevent the matrix $\mathbf{O}$ from losing rank if the velocities in the horizontal plane are parallel, as long as the vertical components are different 5.2.1. However, it can be noticed that C^{-1} reaches in general lower values with respect to the 2D case, ranging from 0.01 to 0.065, suggesting that the 3D system remains on the overall less observable. The curious result is that the minimum in the observability index is reached around $\psi_{ij} = 90°$, meaning when the two drones are flying with orthogonal velocities. This finding is unexpected since intuitively the relative position should not be relevantly ambiguous in this condition. This case will be studied more in detail in the following sections.

Effect of relative vertical position $\mathbf{z}_{ij}$ on the observability index

The second parameter to be tested is the relative vertical position z_{ij}, which is relevant only for the 3D case, since it does not appear in $\mathbf{O}^{2D}$. It is fundamental to evaluate the response of the system to variations in this quantity since it is the only parameter that affects only the 3D system and it is needed for studying the difference in the observability of the two cases in similar conditions, that is to say when the two MAVs belong to the same horizontal plane ($z_{ij} = 0m$). To this end, the 3D system has been simulated for values of z_{ij} ranging from $-6m$ to $6m$, where a positive value indicates that MAVj is flying higher than MAVi and a negative value suggests the opposite.

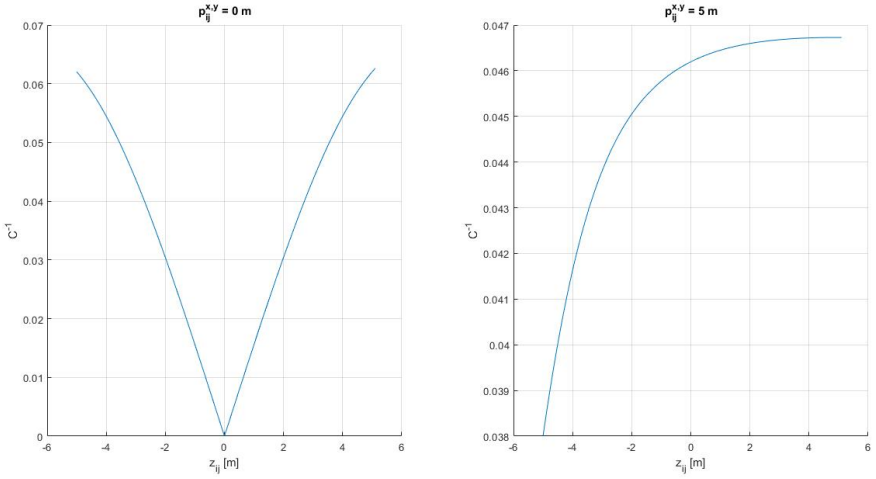


Figure 4.4: *Effect of relative vertical position z_{ij}*

In Figure 4.4, the plots show the trend of the local estimation condition number C^{-1} for different values of z_{ij} when the drones are in the same vertical direction $\mathbf{p}_{ij}^{x,y} = 0$ (left) and when they are not (right). It can be observed that in the first case the observability of the system has a symmetrical course for positive and negative values of z_{ij} around $z_{ij} = 0m$, which corresponds to an unobservable condition as explained in 4.2.1. On the other hand, when the drones do not belong to the same horizontal plane, the trend of C^{-1} is not symmetrical around 0, but it steadily increases as MAVj occupies a position furtherly above

MAVi. This asymmetry will be observed also in other simulations in the following sections, and determines a distribution of the observability different according to the relative vertical position of the drones.

Effect of the relative position on the observability index

In light of the findings of the previous section, it is now possible to fix the values for ψ_{ij} and z_{ij} to study the impact of the relative position in the horizontal plane, of the yaw rates and on the relative velocities on the observability of the system.

Figure 4.5 reports the colormaps obtained by varying the relative (x,y) position of the drone in the range [-5, 5] m and keeping constant all the other parameters as fixed in Table 4.1. The index C^{-1} changes its value as indicated in the colorbar. It is important to pay attention to the fact that the same color in the two maps does not correspond to the same value of the number.
This study has been conducted by considering the MAVs on the same height, in order to appreciate the difference in the behavior under the same conditions.

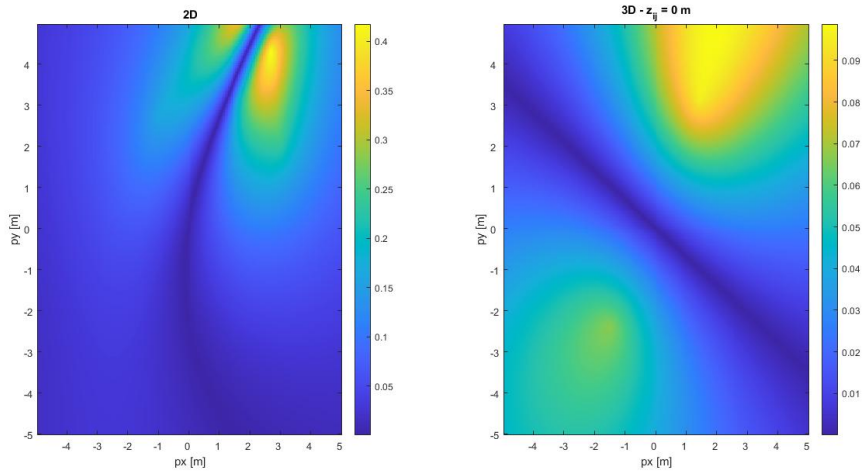
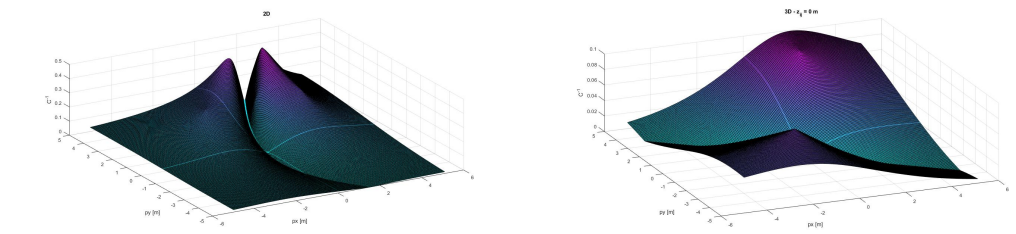


Figure 4.5: *Effect of relative horizontal position*

The main consideration thatcan be done is that in the 2D case C^{-1} reaches higher values with respect to the 3D case: 0.4 and 0.09. The distribution of the observable areas appears to be symmetrical around a line that divides the mapping in two halves: the position of this line varyies according to the parameters that are fixed at the beginning of the analysis and can be shifted, reversed or moved by tuning the relative yaw angle and relative yaw rate.

Effect of velocity of MAVj on the observability index

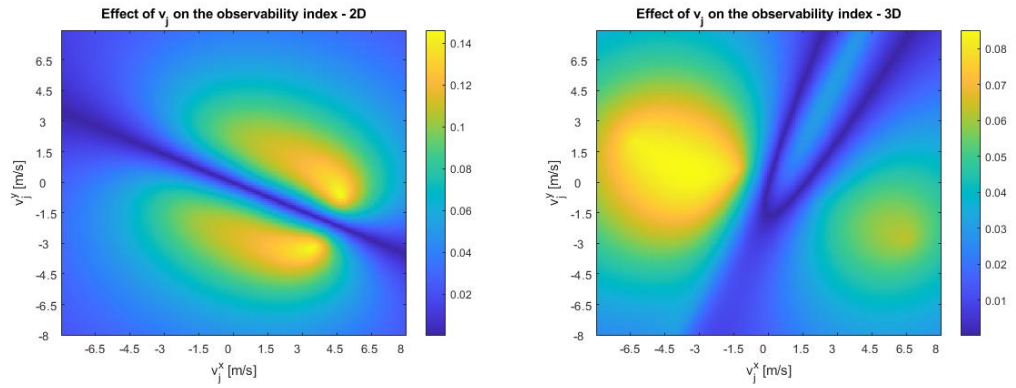
In Section 4.2.1 it has been observed that the velocity of MAVj is a critical parameter: it appears, in fact, that MAVi is not able to track MAVj anymore if the latter is not moving. For this reason, in this section the effect of the velocity of MAVj in the horizontal plane is varied in the range [-8, 8] m/s in order to see how this affects the index C^{-1}. The drones



(a) (b)

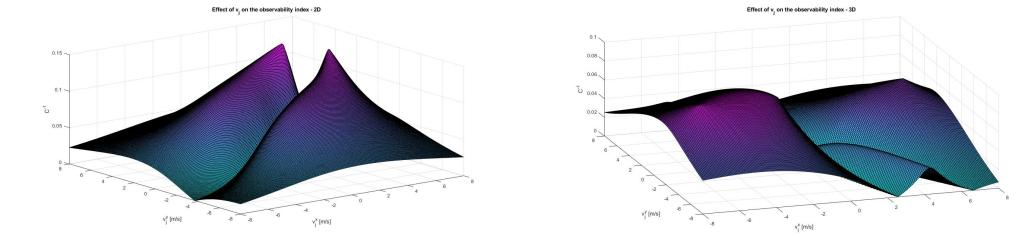
Figure 4.6: *Level maps of C^{-1} under the variation of the relative horizontal position*

are again assumed to belong to the same plane.
The results are shown in Figure 4.7 and 4.8.



(a) (b)

Figure 4.7: *Effect of the velocity of MAVj on C^{-1}*



(a) (b)

Figure 4.8: *Level maps of C^{-1} under the variation of MAVj velocity*

The results are comparable to those found by varying the relative position: the 2D system is more observable and the distribution of the observable conditions shows some symmetry around an unobservable line that depends on other fixed parameters.

Effect of yaw rates on the observability index

In this section, the effect of the yaw rates r_i and r_j is studied. The yaw rates are varied around the initial value $[0, 0]^\circ/s$ in the range $[-90, 90]^\circ/s$ for both the 2D and 3D cases. The results are shown in Figure 4.9 and Figure 4.10, which show the value reached by C^{-1}.

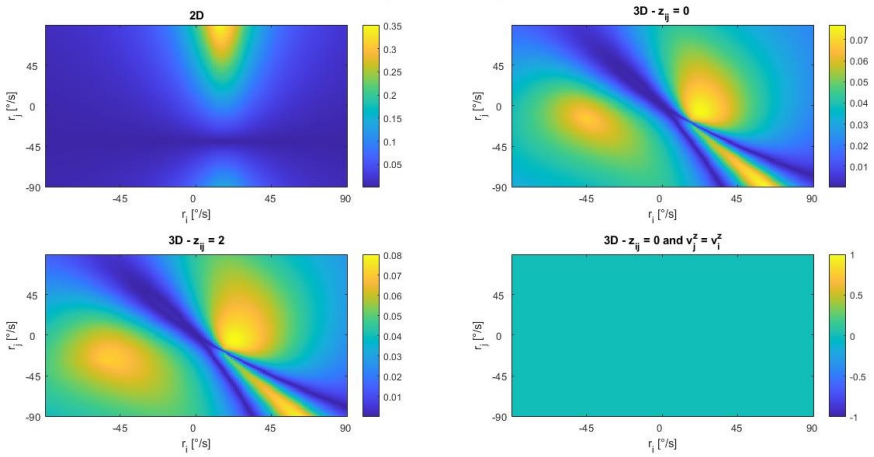
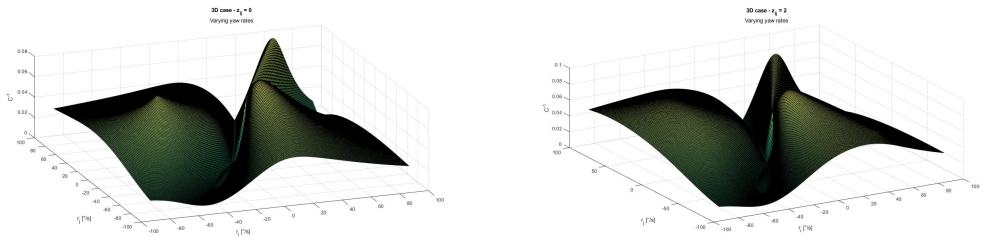


Figure 4.9: *Effect of the yaw rates*



(a) **(b)**

Figure 4.10: *Level maps of C^{-1} under variation of the yaw rates*

As before, the first aspect to be noticed is that the local estimation condition number assumes significantly higher values in the 2D case with respect to the 3D case: 0.35 against 0.07.
Another interesting aspect is that the variation of the relative vertical position affects the observability conditions: it appears that for drones belonging to the same plane the area characterized by higher values of C^{-1} is smaller than that observed for MAVs on different horizontal planes. This observation confirms again the positive effect on the system of guaranteeing the operation on different levels.
Finally, a new test has been proposed to check the unobservability of the system under variations of the yaw rates when one of the unobservable conditions discussed in section 4.2.1 occurs, specifically when $z_{ij} = 0$ m and $v_j^z = v_i^z$. The system results, as expected,

unobservable for any value of r_i and r_j, validating the results of the rank condition check.

Asymmetry due to the relative vertical position

A final test has been run in order to validate the trend of C^{-1} connected to the relative position in the vertical direction of MAVi and MAVj as observed in Section 4.5.2. To this end, the system has been simulated by varying the velocity of MAVj in the horizontal plane (Figure 4.11) and the relative (x,y) position of MAVi and MAVj for different z_{ij} (Figure 4.12), by keeping all the other parameters of the system fixed.
It can be observed that the maximum value of C^{-1} is the same for all the levels studied and equal to 0.08. However, the distribution of the observable conditions appears to be highly affected by the relative vertical position z_{ij}.
By observing Figure 4.11, it can be noticed that the situation where MAVj is found at a lower height with respect to MAVi is beneficial from the observability point of view, resulting into a wider observable area. Moreover, this distribution appears to be inverted if MAVj is instead above MAVi (positive z_{ij}): in the first case, the best values for MAVj velocity are negative in the x direction and positive in the y direction, while in the second case the parameters are switched.

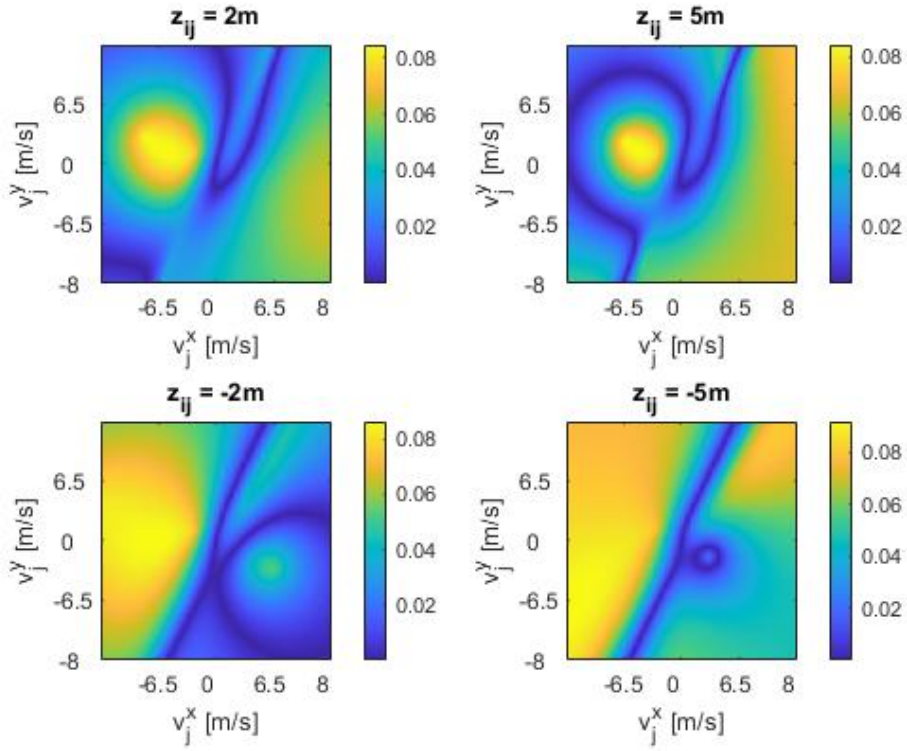


Figure 4.11: *Effect of the asymmetry in the observability of the system due to different relative vertical position of the MAVs - varying relative horizontal velocity*

The same asymmetry is observed also when the relative horizontal position is varied: starting from a situation where MAVj is higher than MAVi (positive z_{ij}), the best conditioned distributions appear to shift as MAVj climbs down under the level of MAVi, moving from negative p_x and p_y for positive p_x and p_y. The distribution of the asymmetry has been observed also to depend on the relative yaw rates. In this case the system has been simulated with $r_i = 10°/s$ and $r_j = -20°/s$: by inverting them, the distribution appears to be reversed.

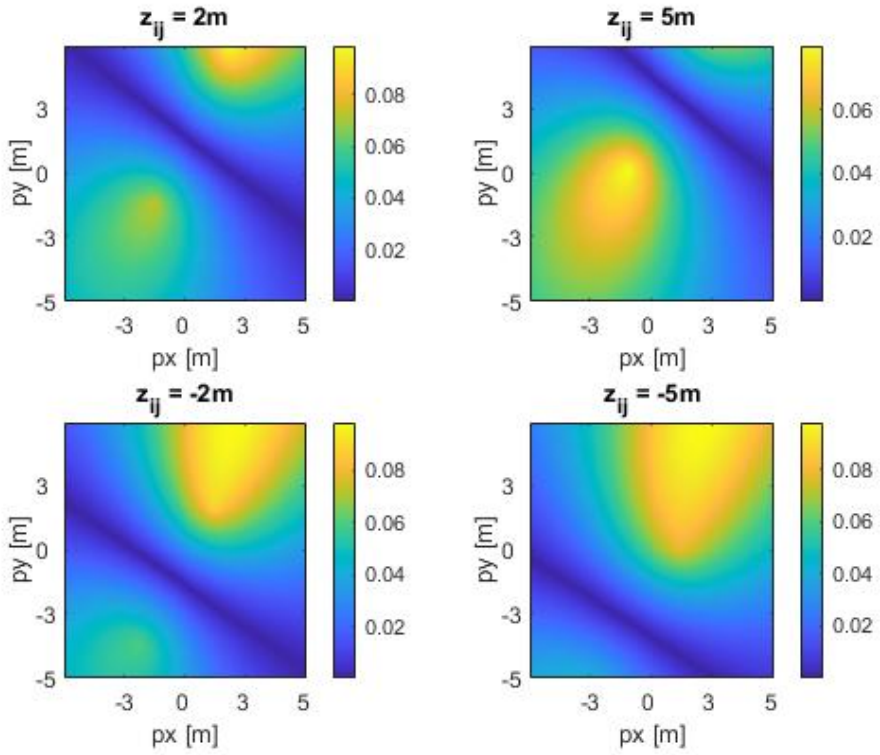


Figure 4.12: *Effect of the asymmetry in the observability of the system due to different relative vertical position of the MAVs - varying relative horizontal position*

4.5.3 Conclusions on the observability analysis through local estimation condition number

On the basis of the results obtained by the examples described, it is possible to drive some conclusions about the degree of observability of the system:

- The 3D system is in general less observable than the 2D one. This is due to the rìgreater difficulty of distinguishing the states to be estimated given the same range and velocity vectors, as the measured outputs can correspond to a higher number of combinations of states and inputs and the solution is not unique (same range and inputs can be associated to a relative position of MAVj belonging to any point of a sphere centered in MAVi and with radius d_{ij}).

- The 3D system guarantees a local weak observability in a higher number of cases, such as coincident (x,y) position and same horizontal velocites.
- The distribution of the observable situations strongly depends on the relative position in the vertical direction, on the relative yaw angle and on the relative yaw rate.

On the overall, the number of parameters that concur in the well-posedness of matrix **O** are numerous and hard to isolate. The multifactorial nature of the problem requires a deeper and more attentive study in order to determine the real impact of every parameter. The one proposed here is just qualitative and shows the most glaring effects.
One final consideration is that the local estimation condition number tìnever overcomes the value of 0.1 in the model presented in this thesis, whil in 2D it is observed to be significantly higher in most observable situations: this observation raises the question about whether this value would be sufficient for deploying the system in real practice. It may be possible that the grade would be too low for guaranteeing convergence of the estimation.

4.6 Verification through simulation

In order to evaluate the correctness of the results obtained through the observability analysis, it is necessary to validate them by simulating the system while keeping in mind the findings from the study. To this end, the system described in Chapter 3, made up by two moving drones, where MAVi attempts to estimate the relative position of MAVj in its own body frame is used.
The main focus of this simulation is to understand the relationship between the value of the local estimation condition number and the convergence time of the EKF in real situations. The parameters that are selected for evaluating the quality of the estimation are the convergence time of the estimation error E, the local estimation condition number C^{-1}, the eigenvalues of the error covariance matrix **P** and the minimum singular value of matrix **O** $min(sing(\mathbf{O}))$. The choice is dictated by the mutual dependence that exists among these quantitie: the value of the smallest eigenvalue of **O** determines a smaller C^{-1}; a small local estimation condition number suggests that the system is not observable and consequently the uncertainty in the estimation is expected to be relevant, resulting into larger eigenvalues of **P** and a longer convergence time of the estimation error. This behavior is what we want to validate by observing it in realistic simulation.
In the following, the results obtained when imposing random trajectories to the drones and when adopting trajectories that avoid on purpose unobservable conditions 4.14 are reported. The ideal conditions imposed in Figure 4.14 aim at being favourable to the estimation performed by the EKF and involve careful design both of the paths followed by the drones and of the velocities to be used. The setup consists of letting MAVi fly along a clockwise circular trajectory of radius ρ at an angular velocity ω at a height z_i, which oscillates in a sinusoidal way. Mavj, on the other hand, follows a circular path concentric to MAVi, counterclockwise, with a radius $\rho + 3$, an angular velocity 2ω and a height $z_j \neq z_i$. With this setup, parallel velocities are avoided, as well as possible hits between the MAVs. Moreover, the difference in the velocity magnitude allows to excite the system with sufficient dynamics.

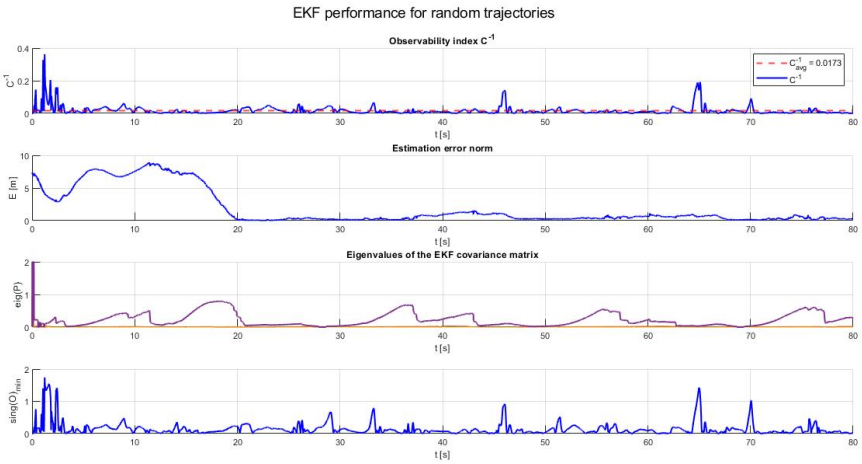


Figure 4.13: *Simulation with random trajectories*

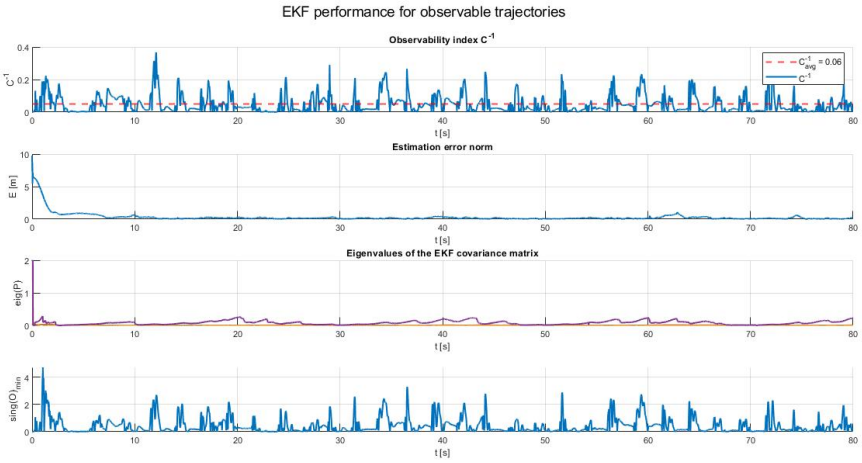


Figure 4.14: *Simulation avoiding unobservable conditions*

As it can be noticed by comparing Figure 4.13 to Figure 4.14 the reults obtained by imposing favourable trajectories are relevant. The average value of the local estimation condition number C_{avg}^{-1} is increased from 0.0173 to 0.06. This is due to the better conditions of matrix $\mathbf{O}$, which can be seen in the higher value of the minimum singular value in Figure 4.14, and to the less uncertainty in the estimation, resulting into lower eigenvalues of matrix $\mathbf{P}$. Finally, the improvement becomes tangible when observing the effect on the convergence time of the system: when the trajectories are random, the estimation error E converges in 20s, while by imposing concentric paths the convergence time is decreased to 10s.

4.6.1 Orthogonal trajectories

This kind of verification has proved useful also from an other point of view. By observing closely figure 4.13, it is possible to see that after the convergence of the filter around 20s, the estimation error increases again to high values between 37s and 47s. This behavior suggests that around those instants some condition occurs that determines a situations where either due to the input velocities of the drone or to their relative position (or a combination of both), such that the system becomes less for long enough to jeopardize the correct convergence of the EKF. In order to understand the reason beyond this behavior, a snapshot of the trajectories of the drones between 35s and 50s has been taken and is shown in figure 4.15.

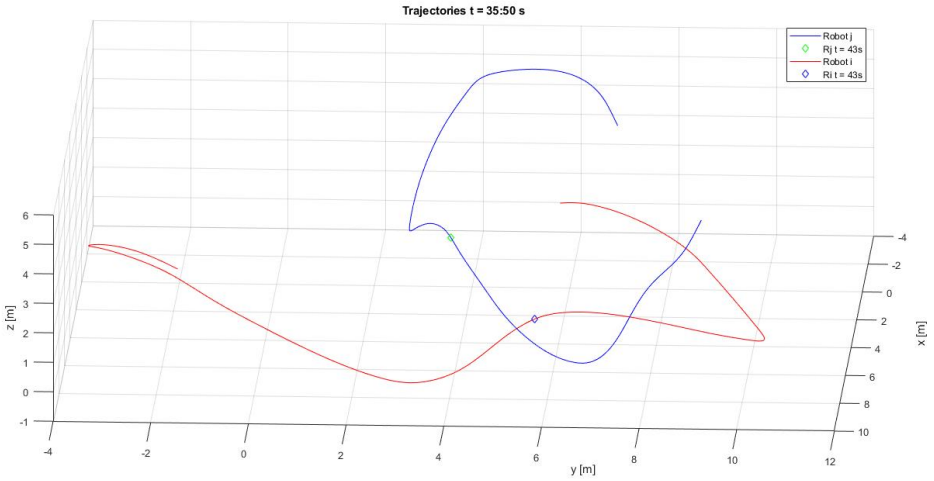


Figure 4.15: *Effect of relative vertical position*

The first observation that can be done is that around 40s the two drones cross each other normally, although on different planes, such that no collision occurs. The fact that orthogonal trajectories may decrease the local estimation condition number of the systems sounds counterintuitive, although evidence has been found already with the observability analysis in Section 4.5.2. Here, in fact, the local estimation condition number appears to drop in the 3D case when ψ_{ij} approaches 90°. The observations from simulation appear to validate this behavior, although it is still to be determined wheter it depends only on the orthogonality of the trajectories or if there are other underlying condtions that are not accounted for and that determine this behavior.
By verifying also the conditioning of the observability matrix $\mathbf{O}$, it has been observed that the determinant decreases significantly when orthogonality is imposed, because the entries in the last column very low values.

4.7 EKF performance evaluation

In light of the results obtained through the observability analysis and its validation through simulation, it is possible to evaluate the performance of the EKF, keeping in mind the effect of states and inputs on the observability of the system.

The focus of interest is on the quality of the estimation provided and on the convergence time, which here indicates the time that the filter requires to reach an acceptably small estimation error.
To this end, the system made up by two MAVs, MAVi (host) and MAVj (tracked), is simulated by imposing various trjectories, some designed to be favourable to the estimation problem, others randomized.

4.7.1 Convergence time

The convergence time of the estimation error computed as the norm of the error on the three states for the relative position in the three dimensions is certainly the most revealing parameter for evaluating the qualituìy of the optimal state estimator.
It is useful to know the expected time that the EKF will take to converge in view of the design of a rela-time simulation. In fact, when deploying the algorithm on real drones it is necessary to include an initialization lapse of time in order to allow the filter to converge before applying the flocking algorithm, so that the robots would be able to correctly localize each other to avoid computing their own trajectory on the basis of incorrect information. Knowing this exact period of time needed on average by the EKF can be helpful to optimize this initialization procedure.
In order to exctract this information, the system is simulated 70 times with randomly assigned initial positions of the drones within a given volume, random initialization of the filter and random initial inputs. The resulting estimation error is shown in Figure 4.16.

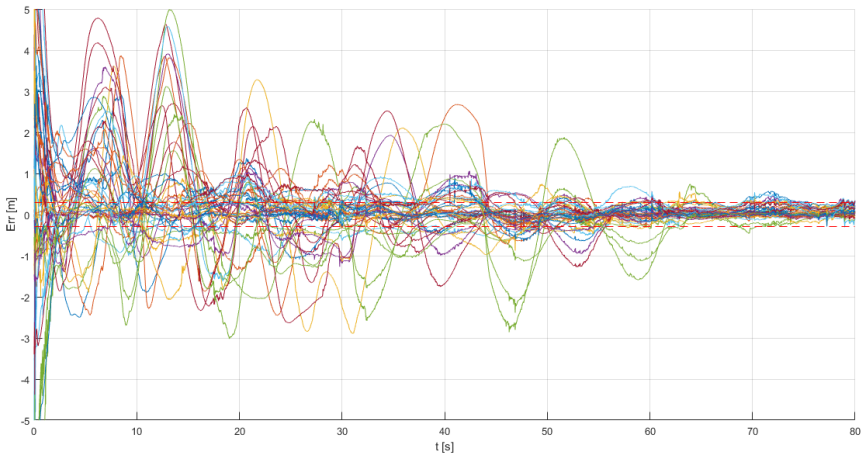


Figure 4.16: *Convergence time - random trajectories*

In order to verify the findings of the observability analysis, a second experiment has been conducted with the same setting as the previous one, with the difference that this time the imposed trajectories are not random anymore, but specifically designed to avoid unobservable maneouvres and to excite the system with enough dynamics to maximize its observability. The designed trajectories are similar to those described in Section 4.6. the result is shown in 4.17.

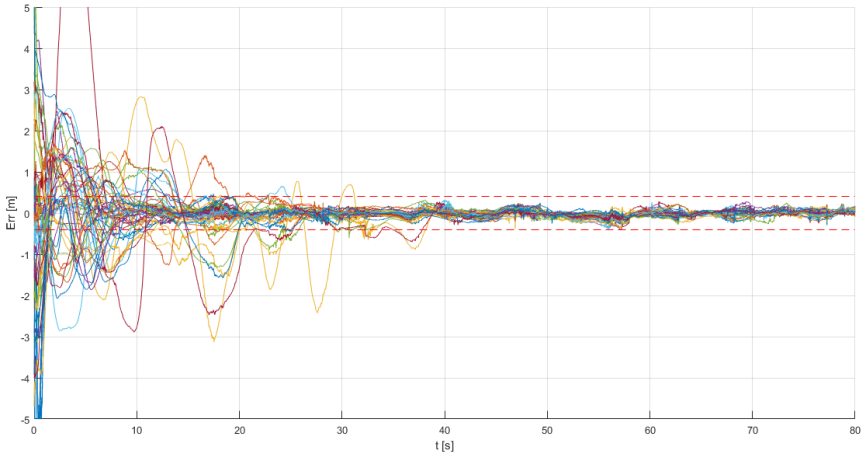


Figure 4.17: *Convergence time - favourable trajectories*

By comparing Figure 4.16 and Figure 4.17 it is possible to see that the application of favourable trajectories relevantly affects the convergence time of the filter.

4.7.2 Sensitivity to measurement noise

A final verification of the EKF robustness has been carried out to evaluate the effect of a noisy range measurement on the ability of the estimator to predict the relative position of MAVj with respect to MAVi. It is especially important to assess the effect of uncertain data on the observation in the problem addressed in the present thesis because it is the only measurement employed to perform the estimation, so uncertainty on this quantity is expected to affect significantly the convergence of the filter. Moreover, from the observability analysis, it results that the system needs information from up to the third Lie derivative and therefore it is predictable that the EKF will see a deterioration in the quality of the estimation as the noise on the range measurement increases.
The experiment is carried out by designing trajectories for the two MAVs that avoid on purpose unobservable conditions: the robots travel on different planes, avoiding parallel trajectories and moving in a sinusoidal way on the vertical direction. Moreover, since from the observability analysis it results that a delta in the the magnitude of the velocity is a favourable condition, MAVj moves at a velocity that is double in module of that of MAVi. In this way, the system is excited with sufficient dynamics to reach a good observability.
In order to avoid the effect of process noise, the variance σ_v and σ_w are set to zero.
Finally, also the states and inputs are initialized to the true values, so that the error due to filter initialization will not affect the output of the experiment.
An increasing value for the measurement noise variance σ_r is applied to the system and the quality of the estimation depending on the noise on the observation is evaluated from a statistical point of view, by simulating the system 100 times for 60s for each standard deviation. At each run, the estimation error is computed by combining the error in the estimation of the three states (relative position $\mathbf{p}_{ij}$ in the three dimensions) as:

$$Err = \sqrt{(err_x^2 + err_y^2 + err_z^2)} \tag{4.19}$$

After 100 runs, the AMAE of the error (Absolute Mean Absolute Error) is computed to understand the average estimation quality for each noise realization and the standard deviation of each mean error MAE (Mean Average Error) is extracted to evaluate its variation. The results are summarized in Table 4.2 and in Figure 4.18.

Range noise (σ_r) (m)	0.1	0.25	0.5	1	2	5	8
AMAE (cm)	2.89	4.27	7.60	22.74	67.00	192.00	290.00
SD (cm)	1.90	3.56	4.00	15.00	36.00	50.00	79.00

Table 4.2: *AMAE and standard deviation for different noise realisations on the measured range*

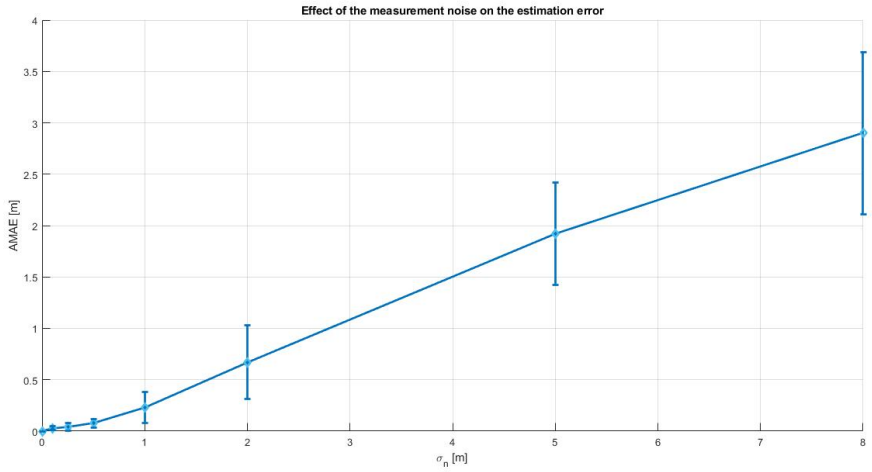


Figure 4.18: *Sensitivity to measurement noise*

Chapter 5

Flocking algorithms for flapping MAVs

In this section, two algorithms for performing fully autonomous flocking with MAVs using only on-board equipment and knowledge about the range are presented.
Both are obtained by applying Reynolds' theory of boids ([34]) and are based on purely range-based local interaction within the flock. The main characteristic that is looked for is the ability of the agents to group up and show an emergent behavior without the need for a predefined trajectory, in order to navigate the environment as a single entity and to reach a consensual direction. The algorithms must also guarantee the avoidance of collisions within the flock and a fast convergence of velocities to a similar magnitude.

The first algorithm aims at developing a self-organizing flocking behavior in a constrained environment, by employing agents with the same set of available information and able to navigate in a limited space thanks to a wall avoidance feature.

The second algorithm embeds a leader-follower task, where one of the agents is assigned with a preferred trajectory that is communicated in cascade through the followers; to guarantee safety within the flock, also separation and collision avoidance rules are applied. Cohesion is not used, because the coherence of the flock is already guaranteed by the allocation of a desired trajectory.

The models have been tested first without the Extended Kalman Filter and the real model of the drones in order to see the effect of each drive's component on the overall look of the flock and a varying number of agents has been simulated to understand how the algorithms performed by scaling the size of the group. For this first step, the simulation environment has been built only in MATLAB.
Once the algorithms were verified to work conceptually well, the EKF for relative localization and the drone model were added. This step required to modify some of the parameters in order to optimize the performance and the building of a more complex environment in MATLAB and Simulink.

5.1 Simulation environment

In this section, the final simulation environment is described.
It has been built in MATLAB and Simulink and it is common to both algorithms. What has been modified to apply the two different procedures is the code inside the MATLAB function that implements the flocking algorithm and the communication protocol among the agents.
The overall structure of the Simulink model is shown in Figure 5.1:

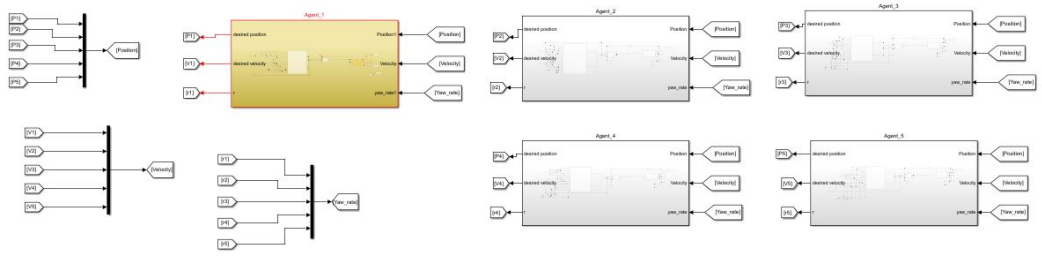


Figure 5.1: *Simulink model - Five agents*

In Figure 5.1 it is possible to see the five subsystems that correspond to each agent in the flock: as it can be noticed, the model has a modular structure, since every drone corresponds to an indipendent system; in this way, it is easy to see that adding or removing a block would not affect significantly the overall system, making the flock fully scalable. The "GoTo" and "From" blocks are needed to move the information coming from the IMU and the UWB around the group: the multiplexers on the left collect all the knowledge about (from top to bottom): **position**, **velocity** and **yaw rate** as measured by the IMU and transformed from the body frame B_i to the horizontal frame H_i, as discussed in Chapter 3. The information is input to each agent's subsytem and is employed for performing the relative position estimation of the other agents and for computing the personal target velocity according to the flocking algorithm. The data about all the agents is spread to the entirety of the flock, although the actual use of this information is decided by the code that implements the flocking algorithm and depends on the topological relative position or on the range between two specific members, which is determined by the working distance of the UWB: depending on the range measured from the other agents, the host drone can decide which rules to apply, if any. For instance, if out of four sorrounding agents, one is further than the maximum communication range, two are closer than the minimum separation range and one is midway between these two radii, then the host will apply only separation from the two closest ones and alignment to the fourth, while the furthest agent will not be considered in the computation of the target velocity.

In 5.1, the first subsystem in yellow corresponds to the so called "agent 1", which is identical to all the other boids in the group in the first algorithm named *Flocking algorithm without leadership*, but it takes on a special role in the second algorithm, referred to as *Flocking algorithm with informed agent*: with this protocol, in fact, the aim is to develop a swarming behavior able to follow approximatively a given trajectory in space; to this end, the "agent 1" acts in this context as an "informed leader", which is the only robot in the

flock that does not compute its own trajectory exclusively as a result of the interaction with the remaining of the flock, but it has a preferred trajectory that has to be communicated in cascade to the followers. The structure of "agent 1" subsystem is identical to that of the other drones, but the code that dictates the computation of the target velocity is modified. The internal structure of the agents' subsystems is shown in 5.2, which displays the model for "agent 2", but it can be considered representative of all the drones in the flock. The blocks can thus be described from left to right, following the flow of information:

- Relative position estimation
- Flocking
- DelFly model

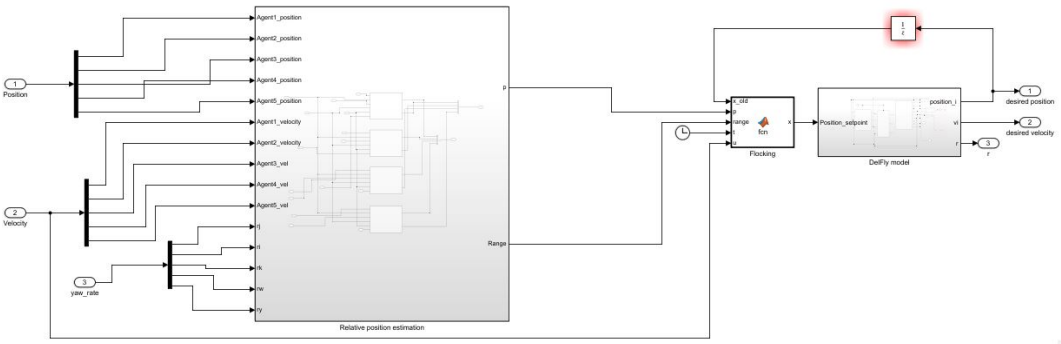


Figure 5.2: *Simulink model - Agent subsystem blocks*

The *Relative position estimation* subsystem collects all the information coming from the demultiplexers on the left and performs the range computation that represents the information coming from the UWB and uses this knowledge to perform the relative position estimation of all the othr agents in the swarm by means of the Extended Kalman Filter described in Chapter 3. The inner model is shown in 5.3: here, it is possible to notice that each subsystem corresponds to one of the other four members of the flock, each one receiving as input its own position, velocity and yaw rates at the previous time step and the same information of the host MAV that is carrying out the observation. The output of the blocks is the range and the relative position in the host MAV body frame. The quantities are collected by two multiplexers on the right and enter the following MATLAB Function *Flocking* (5.2).
The *Flocking* code implements the designed flocking algorithm, by selecting which agents to consider as part of the host MAV's neighborhood on the basis of the computed range and by weighting the components that concur to the computation of the target velocity: separation, cohesion, velocity matching and the leader-follower task in the second algorithm. The target velocity is used for computing the trajectory setpoint of MAVi on the basis of the current position and velocity as computed by the algorithm according to the classical formula

$$x_{k+1} = x_k + v_k T_s \tag{5.1}$$

The computed trajectory is used as setpoint for the DelFly model in the subsystem *DelFly model*: the block contains the model of the drone and the equation for converting the measurements in the body frame to the horizontal frame, as explained in Chapter 3. The current position, velocity and yaw rate measured by the drone are output and are broadcasted through the flock by means of the *GoTo* and *From*, closing the loop.
Figure 5.4 shows the blocks used for the computation of the range, mimicking the UWB information, of the input velocities and of the EKF for relative position estimation.

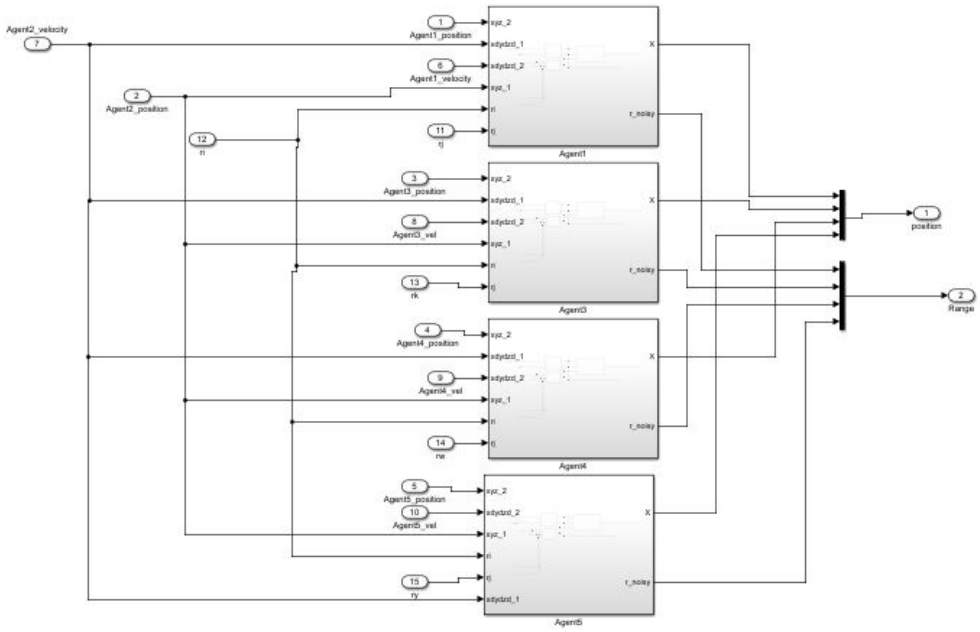


Figure 5.3: *Simulink model - Relative position estimation subsystem*

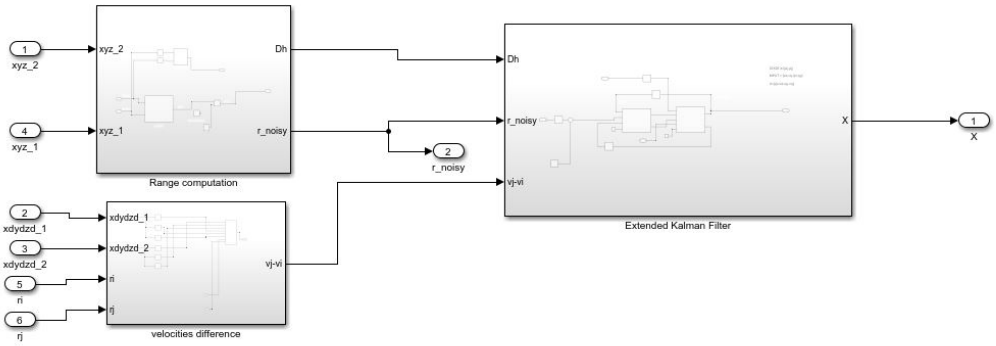


Figure 5.4: *Simulink model - EKF and inputs processing*

5.2 Flocking algorithm without leadership

The first algorithm designed for simulating a swarming behavior is described in the following. The main aim of this solution is to provide a fully emergent behavior suited for indoor applications in a constrained environment. The only additional information needed by the drones other than that described in the previous chapters, is their initial position relative to the walls of an ideal arena. This information would not be needed if the drones were equipped with sensors for obstacle avoidance, as the only purpose is to make them aware of their current distance from the walls.
The flocking algorithm employed in this work is the 3D extension of the model presented in [43] and [41], optimized for the application to the DelFly Nimble and for uncertain knowledge about the relative position of the drones due to the range-only estimation with the EKF. In these works the authors aim at proposing a self-propelled, tunable flocking algorithm. The formulas reported in Section 5.2.1 are partly taken from [43] and [41] and extended to include the third dimension.

5.2.1 Flocking model and simulation parameters

In this section the flocking model is described in terms of mathematical formulation of the algorithm. The target velocity is computed by each drone as the composition of different components that take a specific value depending on the measured range between MAVi and MAVj.
First, the repulsion term is expressed in 5.2. It is applied only when the distance between the two is lower than a certain threshold r_{min}, defined the *Separation radius.* If the drones are safely spaced, this term goes to 0.

$$\mathbf{v}_{ij}^{s} = \begin{cases} c\, r_j \, \dfrac{d_{ij} - r_{min}}{d_{ij}}, & \text{if } d_{ij} < r_{min} \\ 0, & otherwise \end{cases} \tag{5.2}$$

where d_{ij} is the range between MAVi and MAVj and r_j is the relative position of MAVj in the i^{th} frame. c is the separation coefficient.
The resulting escape velocity direction is computed for each drone as the sum of the effects exerted by each drone within its own safe sphere:

$$\mathbf{v}_{i}^{s} = \sum_{j \neq i} \mathbf{v}_{ij}^{s} \tag{5.3}$$

For what concerns the alignment component, it is implemented as a friction force that can be applied depending on the velocity error between the two agents. It has the double function of synchronizing the motion of the flock and of damping the oscillations caused by repulsion. A friction parameter is therefore introduced to weight this component of the velocity, which is much higher than the other weights as this term is responsible for the correct functioning of the flocking behavior, and is fulfills the task of reducing the velocity alignment error. The following equations describe the alignment term. A more detailed description of the principle can be found in [41] and [43].

$$\mathbf{v}_{ij}^{align} = \begin{cases} C_{fric} \, \dfrac{v_j - v_i}{(max(d_{ij}, r_{min}))^2}, & \text{if } d_{ij} < r_c \\ 0, & otherwise \end{cases} \tag{5.4}$$

$$\mathbf{v}_i^{align} = \sum_{j \neq i} \mathbf{v}_{ij}^{align} \tag{5.5}$$

Finally, the effect of the approaching to the walls is accounted for with the shill term. The walls are seen by each agent as a fake agent which is travelling towards the centre of the arena. They affect the trajectory of real drones as another member of the flock would do, by triggering the repulsion term. In this way, when a MAV is flying too close to the walls it is forced to modify its own trajectory in order to avoid them. The same principle could be employed to perform obstacle avoidance.

$$\mathbf{v}_i^{shill} = C_{shill}\, s \left(v_{flock} \frac{x_{shill}}{|x_{shill}|} \; - \; v_i \right) \tag{5.6}$$

where x_{shill} is the distance of MAVi from the center of the arena, $|x_{shill}|$ is the norm and s is a parameter whose value is decided depending on

$$s = \begin{cases} 0, & \text{if } |x_{shill}| \in [0, R] \\ k \, \sin\left(\dfrac{\pi}{d_d}(|x_{shill}| - R) - \dfrac{\pi}{2} \right) + 1, & \text{if } |x_{shill}| \in [R, R + d_d] \\ 1, & otherwise \end{cases} \tag{5.7}$$

where d_d is a safety margin from the wall called the characteristic *width* of the wall.
To the three components described, a fourth one is added in order to provide a self-propelling term able to keep the drone moving even in the case it is separated from the flock.
The final formulation of the target velocity is expressed by 5.8.

$$\mathbf{v}_i^d = \frac{\mathbf{v}_i}{|\mathbf{v}_i|} v^{flock} + \mathbf{v}_i^s + \mathbf{v}_i^{align} + \mathbf{v}_i^{wall} \tag{5.8}$$

By considering the specifics of the drone [20], a maximum speed is introduced so that the desired velocity computed in 5.8 can be saturated at this value in terms of magnitude, while the components in the three directions are kept:

$$\mathbf{v}_i = \frac{\mathbf{v}_i^d}{|\mathbf{v}_i^d|} min\{|\mathbf{v}_i^d|, v^{max}\} \tag{5.9}$$

The values of the parameters in the equations from to 5.9 are obtained by referring to [41] and tuned by trial and error. The final values employed in the simulation are showed in Table 5.1:

Term	Symbol	Value
Sampling time	T_s	0.01 s
Flock velocity	v_{flock}	15 m/s
Maximum flock velocity	v_{max}	15 m/s
Size of the arena	R	80 m
Cohesion radius	r_c	5 m
Separation radius	r_{min}	2 m
Characteristic width of the wall	d_d	0.5
Shill parameter	C_{shill}	10
Shill coefficient	k	0.5
Separation coefficient	c	0.2
Friction parameter	C_{fric}	60

Table 5.1: *Parameters' values for the flocking without leadership algorithm*

5.2.2 Simulation results

In order to evaluate the performance of the flock with the real drone model and the EKF, a simulation lasting 120s is run. The initial positions and velocities are randomly assigned within a range of 10 m and the EKF is initialized with random values. An initialization procedure is used in order to allow the filter to converge before applying the flocking algorithm: the drones fly for 30s following random trajectories in the space, attempting to localize each other in the meantime. Once the 30s are expired, the agents start using the information from the flock mates to compute their own trajectory by applying the algorithm described in Section 5.2.1. The results of this simulation are reported in this section.
The flock is expected to be able to self-organize by collectively select a common direction for all the agents, that tend to match velocities and stay cohesive. When approaching the walls of the arena, the agents should avoid them by picking an escape route and then rejoin in a group.
The resulting behavior of the flock is shown in Figure 5.6: the colored cluster in the middle of the arena corresponds to the random trajectories followed by the agents during the initialization procedure of 30s. After this time lapse, the flocking algorithm is applied and starts computing the target velocity for each drone according to 5.9. As it can be noticed, four agents out of five are able to collectively select one common direction by actively interacting with each other. Only the drone named *agent 2* is separated from the flock. By simulating the system for longer than 120s, it was noticed that this agent was able to rejoin the group after a while, when the others fell again within its radius of sight.
It is interesting to notice how easily the drones are able to pick a new direction when approaching the walls, by turning away from them in a clean way. It is observed that this ability is remarkably improved when the model of the DelFly is introduced, compared to the simulation with the simpler model made up of dots: this attitude can be explained by the specifics of the flapper [20], which is able to navigate with a greater agility compared to common quadrotors. The capability of performing quick turns and more complex maneuvres makes it suitable for rejecting the walls, as it increases the number of possible escape directions. A comparison between the two silmulations can be made by observing Figure 5.5 and 5.6, where the first one shows the simulation with the dots and the second one the full system with EKF for relative localization and the drone model. The system simulated in

5.5 appears to experience greater oscillations when approaching a wall; moreover, the flock tends to stay close to the boundaries for a long time after meeting them, suggesting a kind of bouncing behavior. This can be due to the poor ability of finding an escape rout. On the other hand, the introduction of the DelFly model improves the overall behavior of the flock, both in terms of cohesion and of change in the trajectory to stay away from the walls, as it can be seen in Figure 5.6.

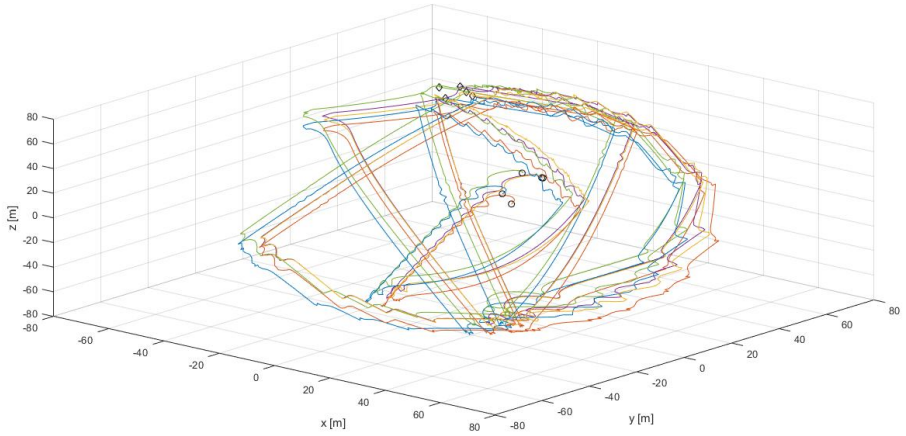


Figure 5.5: *Simulation without EKF and DelFly model*

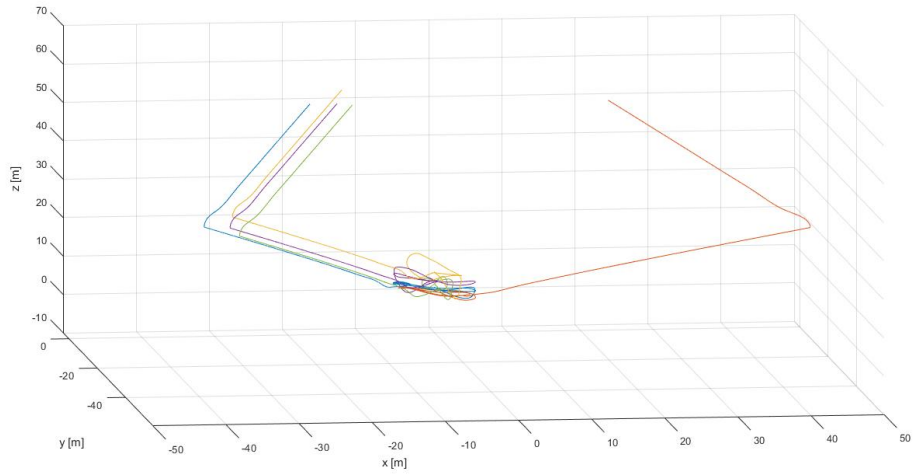


Figure 5.6: *Simulation with EKF and DelFly model*

The simulation shown in Figure 5.6 can be studied more in detail by observing the development of the velocity components of the agents. It is possible to recall that one of the drives imposed to the flock is that of reaching a consensus concerning the selection of a

common velocity, that matches both in magnitude and in direction. In Figure 5.7 the three components of the velocity of each agent are shown: first, in the initial 30s dedicated to initialization of the filter, the agents follow the randomly imposed trajectories; after 30s, the flocking algorithm takes over and the drones are able to promptly find a common direction of motion. The only exception is represented by *agent 2*, which cannot join the flock at the first attempt, but after 105s it can be noticed that it is able to catch up with the flock mates eventually. It is interesting to observe the effect of the approach to the walls on the behavior of the swarm, as it triggers a different response depending on the point at which the multi-robot team finds itself with respect to the borders of the arena.

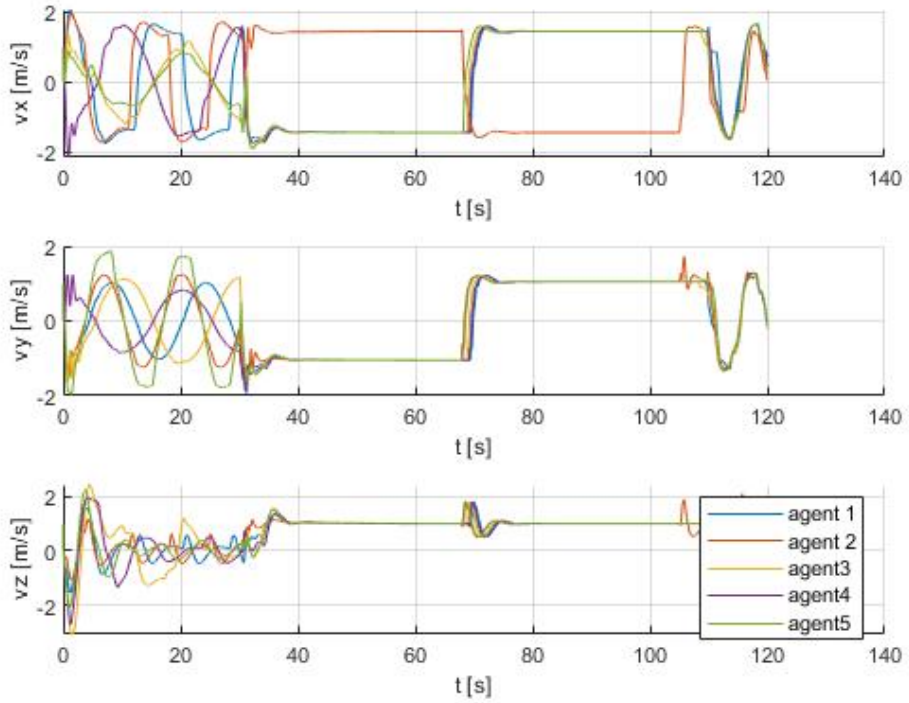


Figure 5.7: *Velocity components - flocking algorithm without leadership*

Figure 5.8 refers to a simulation when no separation took place and shows the magnitude of the velocity of each agent. The first 30s correspond to the initialization procedure for allowing the convergence of the filter, but after this period the agents are able to quickly select collectively a velocity value, which is around the value assigned by the self-propelling term of the algorithm. The vertical dashed lines show the time step at which the agents enter within the radius from the walls that triggers the action of the wall avoidance term: around this instant there is a sharp change in the magnitude of the velocity. During this simulation, the flock comes twice close enough to the boundaries of the arena to be forced to turn away from it. The first occasion occurs around 63s, when the flock approached a wall in its middle: in this case, it is possible to see that the members are able to quickly recover from the change of direction and to match velocities again in a few seconds. The

second situation is observed around 105s and this time it appears that the flock takes longer to organize itself again and the velocities experience a relevant oscillation for over 10s: in this case, it was observed that the swarm approached the border of the arena around one of its upper corners. This situation creates difficult conditions for the selection of a new route, as many escape directions are blocked by the intersection of three walls. One aspect that has been observed to cause this confusion in the presence of a corner is the oscillations of velocities that are probabilbly due to the conflict between the need to avoid the walls and the need to stay cohesive and match velocities with the neighbors. This could be tackled by designing a new algorithm for selecting the escape direction from the walls by, for example, electing a temporary leader that can scan the available getaway angles and drive the flock out of the blockage. A similar suggestion has been expressed in [4] and can be a source of inspiration for a future work.

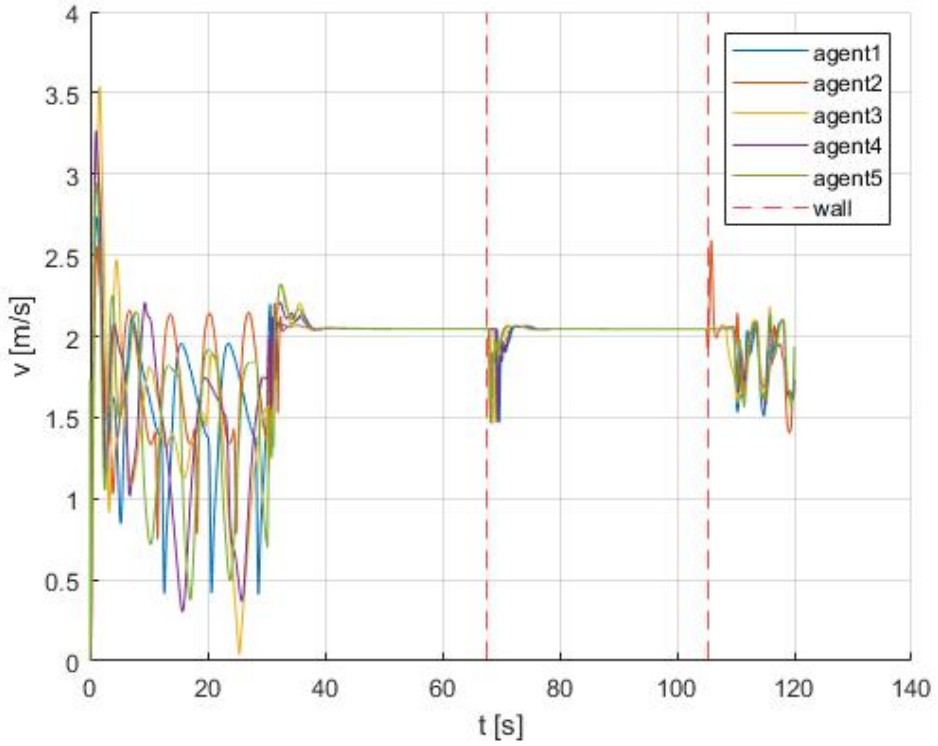


Figure 5.8: *Velocity matching - flocking algorithm without leadership*

5.3 Flocking algorithm with informed agent

The second algorithm that has been designed exploits the theory proposed by Couzin [8], adding to the simple self-organizing behavior a leader-follower feature, that allows the flock to follow a pre-defined trajectory in the space. The solution can be useful to simulate a flock with a natural appearance that, at the same time, can be easily controlled to move in

a desired pattern. The algorithm, as in the previous case, has been tested first without the EKF and the drone model, and later by adding both the state-estimator and the drone.
In order to verify the correct performance of the EKF under different conditions, various trajectories have been designed and tested.

In this section the flocking model is described along with the setting for the simulation. The main source of inspiration for this algorithm is to be found in the Couzin model [8] and the Cucker model [9]. The two algorithms are joined and a leader-follower feature is added, so that the emerging flock is able to follow the trajectory imposed by the leader and concurrently maintain safe conditions within the swarm. The main difference from the first algorithm is that in this case the drones are not confined in a given volume, but the compactness of the flock is achieved by applying the Reynolds' cohesion rule [34] and by adding the leadership feature.

5.3.1 Algorithm design and simulation parameters

The algorithm has been designed for a number N = 5 of MAVs that move in the 3D space. They determine their behavior on the basis of local interaction with neighbors that are able to localize autonomously and which give access to information about their own velocity through communication by means of the UWB antenna. Within the group, only one agent is informed about a predefined trajectory to follow and this knowledge is spread throught the flock in cascade. The need to follow the leader replaces in this algorithm Reynolds' rule of cohesion and it is able to maintain the coherence of the flock by forcing a preferential direction as dictated by the relative position of the preceding drone. The drives that determine the computation of the target velocity for each agent are described in the following and are inspired by the Couzin model ([8]). Each component is then weighted according to the level of priority defined for each task with an approach similar to that employed in the Cucker model ([9]).

Mathematical implementation

First, the main characteristics of the Couzin model are introduced. The fundamental idea is that the boids communicate according to a range-based protocol. The space around each agent can be divided into concentric spheres centered on the boid, where the closest one of radius r_r defines the volume inside which the other agents shall not enter and where the only applied drive is that of separation; the most external one is the cohesion sphere of radius r_c, where cohesion rule is applied; at midway between the two there is the alignment sphere, r_a, where the agents align their velocity to those of their neighbors. Separation rule is the only one existing below r_r, while for ranges $d_{ij} \in [r_r, r_c]$ alignment and cohesion might be applied concurrently.

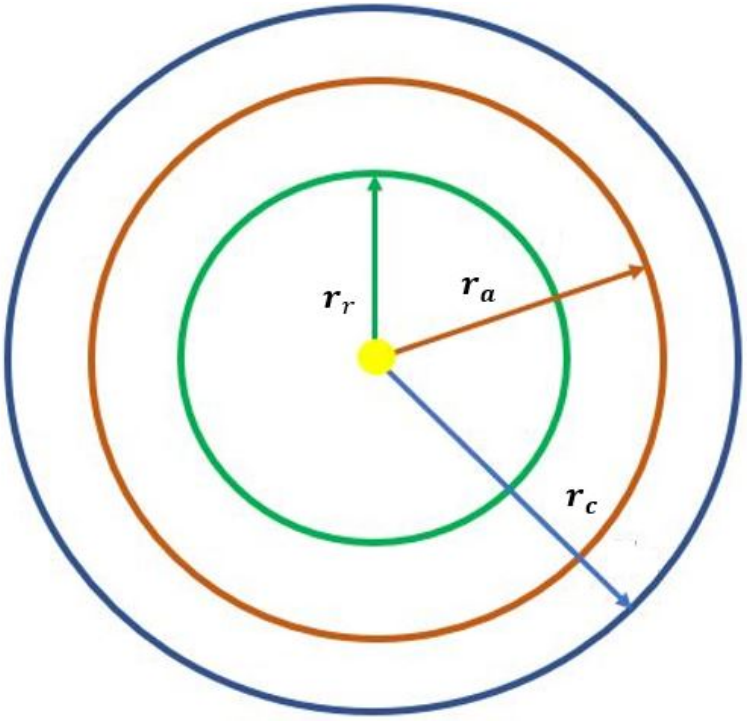


Figure 5.9: *Couzin model - range for separation, alignment and cohesion*

The equations 5.10, 5.11 and 5.12 show the computation of the three components of separation, cohesion and alignment respectively. N_i is the number of boids within sight of agent i, p_j is the position of boid j and p_i is the position of boid i. It can be noticed that separation and cohesion are identical drives, but they impose a reaction in the boid which has opposite direction.

$$S_i(k+1) = -\sum_{b_i \in N_i} \frac{p_j(k) - p_i(k)}{|p_j(k) - p_i(k)|} \tag{5.10}$$

$$K_i(k+1) = \sum_{b_i \in N_i} \frac{p_j(k) - p_i(k)}{|p_j(k) - p_i(k)|} \tag{5.11}$$

$$M_i(k+1) = \sum_{b_i \in N_i} \frac{v_j(k)}{|v_j(k)|} \tag{5.12}$$

In the present work, the cohesion rule is replaced by a leader-follower feature which is implemented by letting each agent communicate its own position to the closest follower, if this is within sight ($d_{ij} \leq r_c$). The communication protocol for the leader tracking task is shown in Figure 5.10.

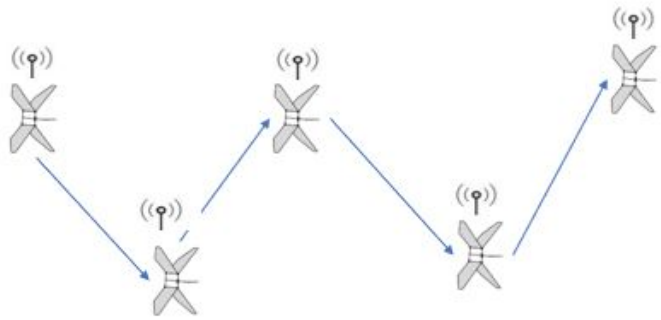


Figure 5.10: *Communication protocol for the leader-follower task*

The computation of the target velocity is performed as a weighted sum of each component, as shown in 5.14. In this algorithm, only the alignment and separation terms are employed, so the component K_i is not used. Therefore, the radii considered are only the repulsion radius r_r and the cohesion radius $r : c$, which in this case corresponds to the maximum communication range and defines the distance at which the alignment drive is applied. *flag* is a flag used to apply only separation when the drones are close to each other and cohesion and leader-follower for ranges greater than the minimum spacing and lower than the maximum communication radius. It is defined as follows:

$$flag = \begin{cases} 0, & \text{if } d_{ij} \leq r_{min} \\ 1, & otherwise \end{cases} \tag{5.13}$$

S, M ang G are the weights assigned to separation, alignment and leader-follower tasks respectively, while $g(k)$ is the relative position of the precedeing drone at time step k.

$$\mathbf{v}_i(k+1) = S\, s(k)\, f + (1-f)\, m\, M(k) + (1-f)\, G\, \frac{g(k)}{|g(k)|} \tag{5.14}$$

The target position is then computed using the kinematic formula 5.16, where v_{agent} is the velocity module assigned to the drone and defined as

$$v_{agent} = min(v_{max}, |v_i(k)|) \tag{5.15}$$

$$\mathbf{x}_i(k+1) = x_i(k) + v_{agent}\, Ts\, \frac{v_i(k)}{|v_i(k)|} \tag{5.16}$$

Table 5.2 shows the values assigned to the parameters of the algorithm.

Term	Symbol	Value
Sampling time	T_s	0.01 s
Flock velocity	v_{flock}	4 m/s
Maximum flock velocity	v_{max}	7 m/s
Weight of the alignment term	m	0.7
Weight of the cohesion term	G	1.4
Weight of the separation term	S	5
Alignment radius	r_a	5 m
Separation radius	r_r	1 m

Table 5.2: *Parameters' values for the flocking with informed agent algorithm*

5.3.2 Simulation results

Just as for the first algorithm presented, also in this case the system has been simulated first with a dot model and later with the DelFly model and the EKF. The difference now is that it is possible to study the performance of the algorithm by evaluating the capability of the flock to follow the trajectory imposed to the leader and in terms of cohesion and velocity matching of the team mates.
Bearing in mind the results obtained from the observability analysis in Chapter 4, different trajectories have been tried in order to test the performance of the relative localization filter under different situations, since, as extensively discussed in 4, the successfull estimation of the states is heavily affected by the inputs and states of the system. The results of these simulations are presented in this section.

Figure 5.11 shows the trajectories followed by the five agents when the system is simulated with the dot model and the EKF estimator. The blue line corresponds to the path of the leader, with the starting point coinciding with the empty dot and the arrival point with the empty diamond. The trajectory imposed in this experiment is that of a helix. It is possible to observe that depending on the initial relative position each drone will receive the information about the trajectory to follow from its predecessor. In fact, in the shown case, the drone labeled as *agent 2*, happens to be the last one reached by the information, which has been spread across the entire flock. For this reason, it can be observed that it is the one whose position differs the most from the leader's one. On the overall, the trajectory imposed appears to be favourable to the observability of the system, by avoiding parallel velocities of the drones and by preventing them from belonging to the same horizontal plane.

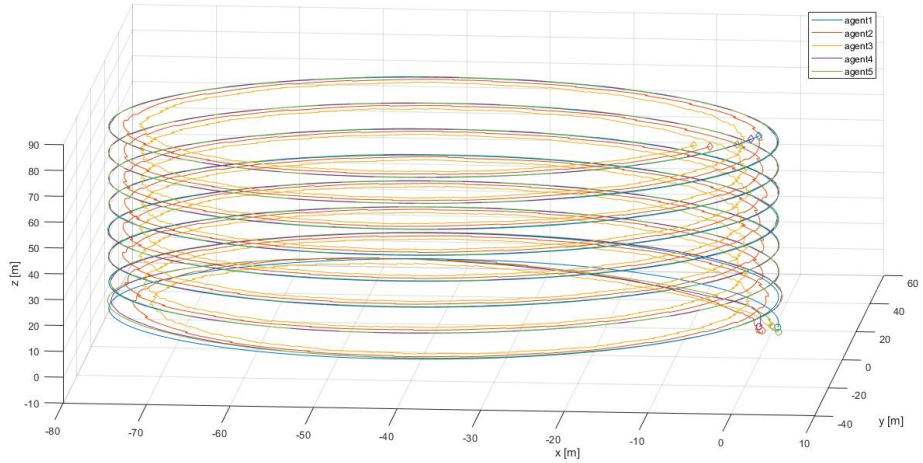


Figure 5.11: *Helix trajectory - simulation with dot model*

Figure 5.12 shows the results of the same simulation in terms of distribution of the flock in the 3D space. Each dot corresponds to one agent and it is possible to appreciate how the flock is able to gradually come together after the 30s of initialization procedure. The top left box shows the position of the drones during te initial phase, when the trajectories are random and they do not apply flocking, such that they are randomly distributed. Starting from the top right box, the agents begin to apply the flocking algorithm. From this point on, it is possible to see that the agents that are not leaders initially group up under the effect of the flocking rule of alignment; then, they start following the leader (blue) and arrange themeself in a queue behind it, reaching the final distribution about 2 min of simulation. The flock is able to maintain the same shape from that moment to the end of the simulation.

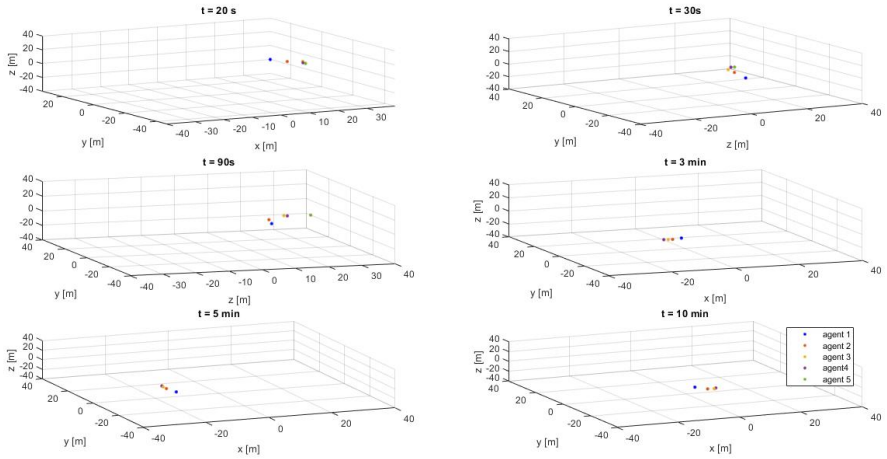


Figure 5.12: *Snapshots of the flocking behavior at different time instants*

The same simulation has been run after integrating the flocking algorithm and the EKF with the DelFly model. This time, the trajectory imposed to the leader was changed from an helix to a more complex shape obtained as a composition of sinusoids:

$$\begin{aligned} v_x &= 5\, cos\frac{t}{10} \\ v_y &= 15\, cos\left(\frac{3\,t}{10} + \frac{\pi}{2}\right) \\ v_z &= 2.5\, cos\frac{\pi}{6} \end{aligned} \tag{5.17}$$

The path followed by the agents is shown in Figure 5.13. It can be noticed that the introduction of the DelFly model and of the EKF can reduce the quality of the performance, however the drones appear to be able to successfully follow the leader along the trajectory.

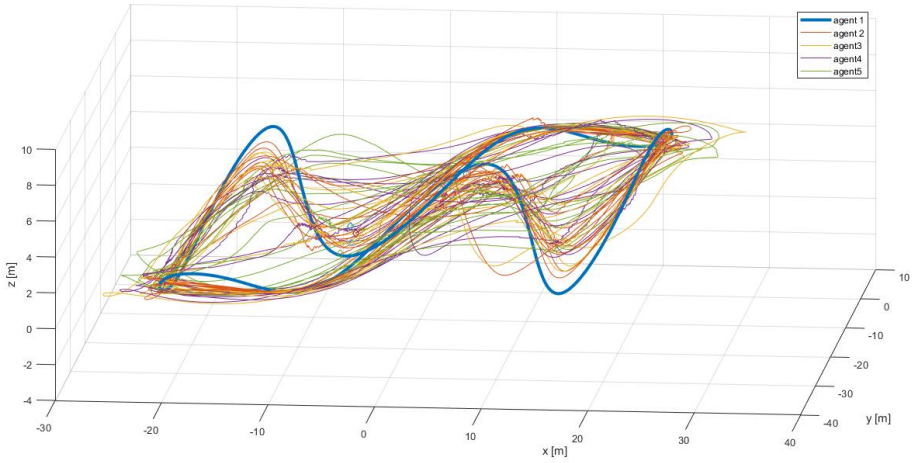


Figure 5.13: *Generic 3D trajectory*

Finally, Figure 5.14 shows the EKF performance for each drone. The estimation is shown for the relative position in the x and y direction of the preceding drone for the four followers, which is actually the only localization needed to apply the leader-follower task. As it can be noticed, the quality if the prediction decreases as agents further along the flock are considered. This is expected because the information about the trajectory to follow is less accurate as it is moved along the flock. Moreover, due to the communication delay between the agents, the followers at the bottom tend to cut corners in order to reach the target position that is communicated by the preceding agent, therefore reducing the dynamics between themselves and the tracked drone.

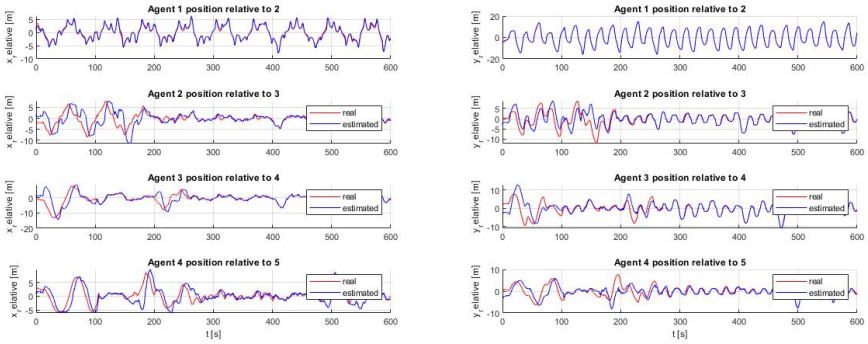


Figure 5.14: *Relative position estimation for the five agents*

Chapter 6

Conclusions and future work

6.1 Conclusions

The study carried out in the present thesis demonstrates the applicability of a fully autonomous, infrastructure-independent, heading-independent, range-based 3D relative localization system for flapping-wing MAVs, which can be employed to perform flocking in GPS-denied, constrained environments with scalable teams .
The 2D filter developed in [26] is proved to work also for 3D applications, therefore broadening the potential implementations in the real world to more complex systems. The designed solution does not use height measurements, but it includes the relative vertical position among the states to be estimated by the observer: this aspect can prove very useful when employing platforms that can provide unreliable height measurements due to complex dynamics, or for application very far from the ground, where it is not possible to have vertical position readings. Moreover, this spares the use of an additional sensor, like down-facing camera or laser-ranger, saving precious load on MAVs, which have already a limited capacity. The method developed overcomes the sensory limits that are required on small payload platforms by exploting only light, commercial and simple pieces of hardware, which makes the work easily replicable for other projects.
One advantageous aspect of the solution lies in its full autonomy: the successfull implementation of a system that relies only on on-board equipment and does not need to refer to a common heading or reference frame can be suited for performing teaming in a ready-to-use manner, with application to heterogenous platforms and in constrained environments. The independence from UWB anchors or GPS allows deployment in indoor and outdoor indistinctly, while the modular structure of the swarm brings a high scalability to the system and robustness to the loss of agents. Finally, the self-organizing nature of the flocking behavior ensures dynamic adaptabily to the environment and freedom from human supervision and guidance.

The subsequent observability analysis validates the overall feasibility of the approach in a punctual manner, by demonstrating the strengths of the method and revealing the weaknesses and limitations of the relative-localization estimator. This study has proved useful not only for analyzing the features of the system, but also for planning the swarming algorithms developed and opens the way to further studies. As a matter of fact, the results from the observability analysis suggest that the swarming behavior requires an attentive planning in order to avoid unobservable situations, which can be effectively handled by designing

flocking algorithms that rely less heavily on the information on the relative position of the other agents in the flock, but can leverage the knowledge about the range coming from the UWB, which can be communicated directly and carries a reasonable amount of noise, without being affected by the observability of the system. A leader-follower task would be more tricky to perform and requires the maintenance of not parallel velocities between the agents of the flock. By joining the leader-follower feature to the application of Reynolds' theory, the advantages from both approached can result in a more stable and reliable demonstration of swarming behavior.

An interesting way for evaluating the performance of the new state observer with respect to the 2D one developed in [26] is to compare the convergence rate in the two situations. In order to perform this experiment, the system made up by two MAVs described in Chapter 3, where MAVi is performing the estimation (host) and MAVj is the observed drone (tracked), is simulated 50 times for 80s. The three situations that are considered are:

- 2D set-up: MAVi and MAVj have knowledge about their height and MAVi attempts to estimate the relative position of MAVj in the plane (x,y directions)
- 3D set-up described in Chapter 3: the trajectories imposed try to avoid unobservable conditions
- 3D set-up described in Chapter 3: the trajectories imposed are random

On every run the convergence time of the Extended Kalman Filter is recorded: here it is defined as the time after which the estimation error, considering the combination of the error in the estimation of the relative position in the three directions, reaches a value between 0.1 m and -0.1 m and remains within this belt until the end of the simulation. The convergence rate indicates in this work the percentage of runs that have reached convergence at each time step, from the starting of the simulation to its end. It is a telling parameter to understand how fast the filter is at estimating the states and what is the average convergence time. This information has a relevant impact on the design of the setup for real-world applications, as it dictates the time needed for initializing the system and can be helpful in order to realize the effective autonomy in terms of battery availability after the initialization.
Figure 6.1 compares the convergence rates obtained through the simulation of the system that performs relative localization in 2D (red) and of the one in 3D, when excited with trajectories that favour the observability of the system (yellow) and random ones (blue).
The results obtained suggest that the 2D filter works remarkably better, as most of the simulations converge within the first 20s and by the 80s all of the runs have converged. On the other hand, the 3D filter has an overall slower convergence, that is not guaranteeed for all of the simulations and for a small percentage of runs (12%) it is never reached. This behavior is explained by observing the values of the observability index introduced in Chapter 4, the local condition estimation number, which reaches values always lower in the 3D case compared to the 2D one. This is due to the higher ambiguity in determining the true relative position of the other drone when adding the third dimension and relying on the same set of inputs, in fact with this set up the same combination of inputs and state can correspond to a greater number of undistinguishable outputs, as the same parameters may refer to any position of the robot on a spehere, while in 2D this uncertainty was limited to a plane.
On average, the convergence time for the 3D case is expected to be around 40s for trajectories that avoid unobservable conditions, but it can be increased to 60s when less favourable

settings are imposed. These aspects underline the need for a precise study of the behavior of the filter under different circumstances and calls for a deeper analysis of the effect of each parameter on the observability of the system.

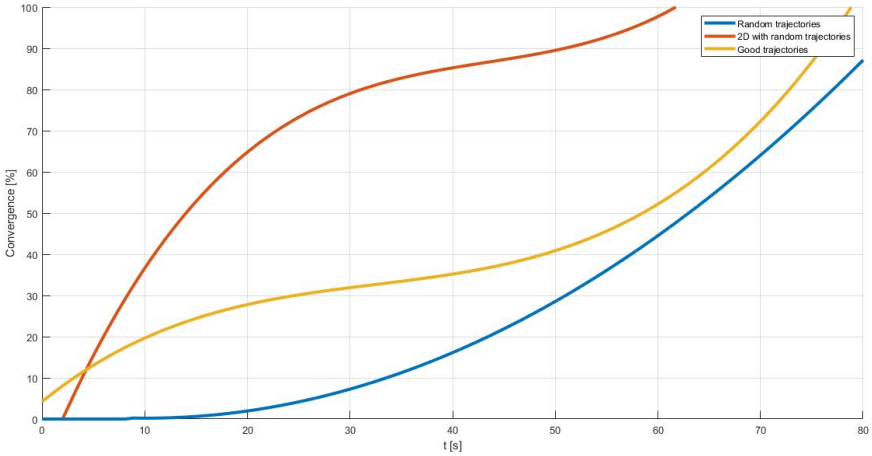


Figure 6.1: *Convergence rate with 2D and 3D relative position estimator*

Finally, the flocking algorithms designed offer a valid solution for performing swarming behavior in constrained environments. They have been specifically designed for indoor application, but the same concepts can be potentially extended to outdoor implementation. With the solution that does not require a leader, the model presented in [41] has been extended to a three-dimensional case and optimized to be applied to flapping wing MAVs. The DelFly Nimble model has shown to be very efficient when approaching the walls thanks to the quick avoidance maneuvres that it can perform. The system has shown a remarkable robustness to the splitting, as by continuing the simulation all of the members are able to rejoin in a cohesive group after separation.
The second algorithm joins a leader-follower feature to the application of Reynolds' rules of flocking, taking advantage from both and resulting into a system able to follow a predefined path in the three dimensional space, successfully avoiding internal collisions a separation of the swarm.

6.2 Future work

There is plenty of envisioned applications that can be derived from the findings presented in this work.
For a start, the relative localization procedure proposed has the peculiarity of being extremely flexible and adaptable and it can be employed in a variety of environments and situations. It would be interesting to test the algorithms in real-world situations and to adapt them to address actual problems. Some of the most interesting applications could be found in precision agriculture, for gathering data for damage or production assessment in fields; the wall avoidance feature of the first algorithm can be easily extended to implement also an obstacle avoidance, useful to perform exploration of inaccessible environments, like

wrecked buildings; finally, teams of coordinated MAVs could be exploited to speed up the complexion of tasks such as mapping of large areas or inspection.

A deeper analysis could be done concerning the observability of the EKF in order to identify other note-worthy situations that can jeopardize the correct convergence of the estimation. As a matter of fact, the observability of the systems is a multi-factorial problem and the value for the local estimation condition number can be the results of the influence of many conditions that are not simple to single out. For this reason, it would be auspicable to provide a more profound evaluation of the effect of the different parameters and their combination. Moreover, the finding concerning orthogonal trajectories raised in Section 4.6.1 does not have a final explanation, as it is corroborated by evidence in more steps of the analysis, but it has not been justified by intuitive understanding. There could be underlining conditions that determine this behavior which have not been accounted for or there might be other reasons that could explicate those observations. For sure, the study done on the observability has proved interesting for predicting what to expect from the filter and can be potentially explored indefinitely for enhancing the overall performance of the filter.

One interesting improvement that can be suggested concerning the flocking algorithms would be to employ a machine learing approach to the problem, like reinforcement learning methods for data aggregation, near-optimal computations and online trajectory reconfiguration. Some examples can already be found in literature, like in [39], where the authors employ deep reinforcement learning (DRL) to perform flocking in complex environments with dynamic obstacles. It would also be auspicable to apply distributed control strategies in order to ensure efficient task allocation for the team of UAVs and guarantee consensus and autonomy.

Unfortunately, the measures for containing the outbreak of Covid-19 made it impossible to test the system on a real platform in laboratory due to limitations in the personnel allowed inside the universities and to restrictions to mobility. Consequently, one natural follow-up of this thesis would be to test the relative localization preocedure and the flocking algorithms on the real drones. This step would be fundamental in order to verify the consistency of the results obtained through simulation and to tune the filter's parameters to real values. Moreover, it would be essential to check the robustness of the EKF to noise on the measurement, which, as explained in Chapter 4, is a crucial point, as the range is the only measurement used for performing the estimation and the system requires information up to the third Lie derivative, therefore uncertainties on this observation could affect critically the convergence of the filter.
In this framework, it would also be possible to try and scale up the team of drones to validate the findings with larger flocks and the robustness and scalability of the algorithms. To this end, basing on the considerations concerning swarming with informed agents in [8], it could be expected that with larger teams the complexity of the problem would not increase significantly, as the percentage of leaders decreases with the size of the flock. Some concern could derive from the communication within the group for larger numbers of agents, so another starting point for improvement could lie in the development of more effective communication algorithms, that could be experimented both leveraging a topological communication among the M closest agents, or a range-based approach like the one employed in this thesis. The target would be to produce a faster broadcasting of the information within the team of robots to compensate for the delays due to the physical transfer of the signals and enhance

the overall performance of the flock.
Finally, it would be interesting to tackle the problem of autonomous realtive localization in 3D by using different combinations of measurements and inputs, according to the sensory equipment available to the elected platform. The one designed in this thesis has been dictated by the choice of the DelFly Nimble as robot model, but the application to other robots could result into a different design of the estimator.

Bibliography

[1] Afzal, Muhammad & Renaudin, Valérie & Lachapelle, Gérard. (2010). Assessment of indoor magnetic field anomalies using multiple magnetometers. 525-533.

[2] A. J. Krener and K. Ide, "Measures of unobservability," Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference, 2009, pp. 6401-6406, doi: 10.1109/CDC.2009.5400067.

[3] Arrichiello, F.; Antonelli, G.; Aguiar, A.P.; Pascoal, A. An Observability Metric for Underwater Vehicle Localization Using Range Measurements. Sensors 2013, 13, 16191-16215. https://doi.org/10.3390/s131216191

[4] Balázs Boldizsár, Vásárhelyi Gábor and Vicsek Tamás 2020Adaptive leadership overcomes persistence–responsivity trade-off in flockingJ. R. Soc. Interface.172019085320190853

[5] Barbara F La Scala, Robert R Bitmead, and Matthew R James. Conditions for stability of the extended kalman filter and their application to the frequency tracking problem. Mathematics of Control, Signals and Systems, 8(1):1–26, 1995.

[6] Brambilla, M., Ferrante, E., Birattari, M. et al. Swarm robotics: a review from the swarm engineering perspective. Swarm Intell 7, 1–41 (2013)

[7] Coppola, Mario & Mcguire, Kimberly & Scheper, Kirk & Croon, Guido. (2018). On-board communication-based relative localization for collision avoidance in Micro Air Vehicle teams. Autonomous Robots. 42. 10.1007/s10514-018-9760-3.

[8] Couzin, Iain & Krause, Jens & Franks, Nigel & Levin, Simon. (2005). Effective leadership and decision-making in animal groups on the move. Nature. 433. 513-6. 10.1038/nature03236.

[9] Cucker, Felipe & Smale, Steve. (2007). Emergent Behavior in Flocks. Automatic Control, IEEE Transactions on. 52. 852 - 862. 10.1109/TAC.2007.895842.

[10] F. Cucker and J. Dong, "Avoiding Collisions in Flocks," in IEEE Transactions on Automatic Control, vol. 55, no. 5, pp. 1238-1243, May 2010, doi: 10.1109/TAC.2010.2042355.

[11] Golub, G.; Loan, C.V. *Matrix Computations*, 3rd ed.; The Johns Hopkins University Press: Baltimore, MD, USA, 1996.

[12] Gu, Y., Seanor, B., Campa, G., Napolitano, M. R., Rowe, L., Gururajan, S., et al. (2006). Design and flight testing evaluation of formation control laws. IEEE Transactions on Control Systems Technology, 14(6), 1105–1112.

[13] Guo K, Qiu Z, Meng W, Xie L, Teo R. Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments. International Journal of Micro Air Vehicles. September 2017:169-186.

[14] Hauert, Sabine & Leven, Severin & Varga, Maja & Ruini, Fabio & Cangelosi, Angelo & Zufferey, Jean-Christophe & Floreano, Dario. (2011). Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. IEEE International Conference on Intelligent Robots and Systems. 5015-5020. 10.1109/IROS.2011.6095129.

[15] Helm, Steven & Coppola, Mario & Mcguire, Kimberly & Croon, Guido. (2020). On-board range-based relative localization for micro air vehicles in indoor leader–follower flight. Autonomous Robots. 44. 10.1007/s10514-019-09843-6.

[16] Hermann, R. and A. Krener. "Nonlinear controllability and observability." IEEE Transactions on Automatic Control 22 (1977): 728-740.

[17] Hwang, M., Seinfeld, J.H. Observability of nonlinear systems. J Optim Theory Appl 10, 67–77 (1972). https://doi.org/10.1007/BF00934972

[18] Kalman R. E., "On the General Theory of Control Systems", Proc. 1st Int. Cong. of IFAC, Moscow 1960 1481, Butterworth, London 1961.

[19] Kajak, Karl & Karásek, Matěj & Chu, Q. & Croon, Guido. (2019). A minimal longitudinal dynamic model of a tailless flapping wing robot for control design. Bioinspiration & Biomimetics. 14. 10.1088/1748-3190/ab1e0b.

[20] Karásek, Matěj & Muijres, Florian & De Wagter, Christophe & Remes, Bart & Croon, Guido. (2018). A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. Science. 361. 1089-1094. 10.1126/science.aat0350.

[21] K. Y. Ma, P. Chirarattananon, S. B. Fuller, R. J. Wood, Science 340, 603–607 (2013).

[22] Konrad Reif, Stefan Gunther, Engin Yaz, and Rolf Unbehauen. Stochastic stability of the discrete-time extended kalman filter. Automatic Control, IEEE Transactions on, 44(4):714–728, 1999.

[23] Kownacki, Cezary & Oldziej, Daniel. (2016). Fixed-Wing UAVs Flock Control through Cohesion and Repulsion Behaviours Combined with a Leadership. International Journal of Advanced Robotic Systems. 13. 10.5772/62249.

[24] Kushleyev, A., Mellinger, D., Powers, C. et al. Towards a swarm of agile micro quadrotors. Auton Robot 35, 287–300 (2013).

[25] Lebar Bajec, Iztok & Zimic, Nikolaj & Mraz, Miha. (2007). The computational beauty of flocking: Boids revisited. Mathematical and Computer Modelling of Dynamical Systems - MATH COMPUT MODEL DYNAM SYST. 13. 331-347. 10.1080/13873950600883485.

[26] Li, Shushuai & Coppola, Mario & De Wagter, Christophe & Croon, Guido. (2020). An autonomous swarm of micro flying robots with range-based relative localization.

[27] Mathworks channel https://www.youtube.com/watch?v=ul3u2yLPwU0

[28] M. Keennon, K. Klingebiel, H. Won, A. Andriukov, "Development of the Nano Hummingbird: A Tailless Flapping Wing Micro Air Vehicle." Paper presented at the 50th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition (2012).

[29] M Boutayeb, H Rafaralahy, and M Darouach. Convergence analysis of the extended kalman filter used as an observer for nonlinear deterministic discretetime systems. Automatic Control, IEEE Transactions on, 42(4):581–586, 1997.

[30] M. Strohmeier, T. Walter, J. Rothe and S. Montenegro, "Ultra-Wideband Based Pose Estimation for Small Unmanned Aerial Vehicles," in IEEE Access, vol. 6, pp. 57526-57535, 2018, doi: 10.1109/ACCESS.2018.2873571

[31] Nijmeijer, H,; van der Schaft, A. *Nonlinear Dynamical Control Systems*; Springer: Berlin, Germany, 1990

[32] N. Trawny, X. S. Zhou, K. Zhou and S. I. Roumeliotis, "Interrobot Transformations in 3-D," in IEEE Transactions on Robotics, vol. 26, no. 2, pp. 226-243, April 2010, doi: 10.1109/TRO.2010.2042539.

[33] Nikolas Trawny, X. S. Zhou, K. X. Zhou and S. I. Roumeliotis, "3D relative pose estimation from distance-only measurements," 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007, pp. 1071-1078, doi: 10.1109/IROS.2007.4399075.

[34] Reynolds, C. W. (1987) Flocks, Herds, and .Schools: A Distributed Behavioral Model, in Computer Graphics, 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.

[35] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. Journal of basic Engineering, 83(3):95–108, 1961.

[36] Rudolph Emil Kalman et al. A new approach to linear filtering and prediction problems. Journal of basic Engineering, 82(1):35–45, 1960.

[37] Saska, M., Vakula, J., & Preucil, L. (2014). Swarms of micro aerial vehicles stabilized under a visual relative localization. In 2014 IEEE international conference on robotics and automation (pp. 3570–3575)

[38] Schranz, M., M. Umlauft, M. Sende and W. Elmenreich. "Swarm Robotic Behaviors and Current Applications." Frontiers in Robotics and AI 7 (2020): n. pag.

[39] P. Zhu, W. Dai, W. Yao, J. Ma, Z. Zeng and H. Lu, "Multi-Robot Flocking Control Based on Deep Reinforcement Learning," in IEEE Access, vol. 8, pp. 150397-150406, 2020, doi: 10.1109/ACCESS.2020.3016951

[40] Vásárhelyi, G., Virágh, C., Somorjai, G., Tarcai, N., Szörényi, T., Nepusz, T., & Vicsek, T. (2014). Outdoor flocking and formation flight with autonomous aerial robots. In 2014 IEEE/RSJ international conference on intelligent robots and systems (pp. 3866–3873).

[41] Vásárhelyi, Gábor & Virágh, Csaba & Somorjai, Gergő & Nepusz, Tamás & Eiben, A. & Vicsek, Tamás. (2018). Optimized flocking of autonomous drones in confined environments. Science Robotics. 3. eaat3536. 10.1126/scirobotics.aat3536.

[42] Vicsek, Tamás & Czirók, András & Ben-Jacob, Eshel & Cohen, Inon & Sochet, Ofer. (2006). Novel Type of Phase Transition in a System of Self-Driven Particles. Physical Review Letters. 75. 10.1103/PhysRevLett.75.1226.

[43] Virágh, Csaba & Vásárhelyi, Gábor & Tarcai, Norbert & Szörényi, Tamás & Somorjai, Gergő & Nepusz, Tamás & Vicsek, Tamás. (2013). Flocking algorithm for autonomous flying robots. Bioinspiration & Biomimetics. 9. 10.1088/1748-3182/9/2/025012.

[44] Y. Cao, M. Li, I. ŠVogor, S. Wei and G. Beltrame, "Dynamic Range-Only Localization for Multi-Robot Systems," in IEEE Access, vol. 6, pp. 46527-46537, 2018, doi: 10.1109/ACCESS.2018.2866259.