AIR FORCE ACADEMY

Target tracking control system for multi-UAV maritime applications

Miguel Guedes Félix

ALF/ENGEL 139928-E

Thesis to obtain the Master of Science Degree in

Military and Aeronautical Sciences - Electrotechnical Engineering

Supervisor(s): Major Tiago Miguel Monteiro de Oliveira
Capitão Diogo Alexandre Oliveira Silva

Examination Committee

Chairperson:	Major-General Rui Fernando da Costa Ferreira
Supervisor:	Major Tiago Miguel Monteiro de Oliveira
Member of the Committee:	Professor Doutor Pedro Tiago Martins Batista

Sintra, October 2022

“Success is stumbling from failure to failure with no loss of enthusiasm.”

Acknowledgments

Com o terminar desta etapa, resta-me agradecer a todos os que tornaram possível a concretização deste projeto.

À Academia da Força Aérea, pela transmissão de valores e conhecimentos essenciais, que me moldaram e permitiram alcançar o que ambicionava. Camaradas e civis, a todos os que cruzaram os seus caminhos com o meu, um bem hajam.

Ao Major Tiago Oliveira, um especial e incansável agradecimento por toda a disponibilidade e sacrifício pessoal que demonstrou durante a realização deste projeto. Os seus conhecimentos, a sua intuição e a sua experiência permitiram ultrapassar muitos dos obstáculos que naturalmente foram surgindo. Será sempre um exemplo a seguir, a nível militar, científico e pessoal.

Ao Capitão Diogo Silva, por toda a sua disponibilidade, pela transmissão de conhecimentos técnicos e pelas discussões de ideias sobre a implementação prática deste trabalho, os meus sentidos agradecimentos. Ao Alferes João Alves, pela ajuda numa fase inicial de implementação e cujo contributo de muito valeu para a conclusão deste projeto, o meu sincero obrigado.

Aos meus amigos e camaradas de curso, os **HAPAXES**, por me terem acompanhado durante toda a minha formação, auxiliando a conclusão da mesma. Por realçarem a importância da lealdade, amizade, companheirismo, por todo o carinho que têm por mim, fico eternamente agradecido. Que voemos juntos durante muitos mais anos.

À minha família, em particular à minha mãe, pelo esforço e dedicação inigualável e pelo seu forte apoio logístico que permitiram o meu foco e concentração neste trabalho. Ao meu pai e irmão, por todo o apoio pessoal. O vosso empenho definiu o meu caminho e resultou no alcançar dos meus objetivos. Espero que se sintam tão orgulhosos como eu me sinto agradecido.

Aos meus amigos, em especial ao Luís, ao Nuno, ao Pedro, ao Miguel e à Raquel, um obrigado pela demonstração de confiança, pelos momentos de fraternidade e descompressão, também essenciais para aclarar ideias e ganhar a inspiração necessária para realizar este projeto.

À Rita Rosado, pelo apoio incondicional e por todo o amor e carinho partilhado. Pela inspiração e motivação transmitidas pelos seus gestos e palavras, ficarei eternamente agradecido. Espero poder acompanhar todo o seu percurso com o mesmo apoio e amor com que acompanha o meu.

Resumo

A importância das aplicações de aeronaves não tripuladas tem sido cada vez mais evidente com o passar dos anos. Enquanto que muita investigação já foi feita no passado, ainda há muitas áreas de desenvolvimento, tópicos e aplicações por explorar e revelar. O Centro de Investigação da Academia da Força Aérea (CIAFA) tem vindo a desenvolver uma nova arquitetura modular de controlo de aeronaves não tripuladas que permite a re-utilização de módulos de controlo de trajetória, visão computacional, aprendizagem automática e fusão de dados sensoriais aplicáveis a um espetro alargado de missões, que inclui, entre outros, aterragem automática ou seguimento de alvos terrestres com base em visão computacional.

Esta tese contribui para a implementação formal de uma arquitetura de comando e controlo, a ser utilizada em missões de busca e seguimento de alvos em contexto marítimo. Essa arquitetura foi idealizada em Santos (2021), recorrendo a 3 modos de operação principais: modo de busca, modo de seguimento e modo de anticolisão. Em particular, esta tese contribui para a implementação do modo de seguimento dessa arquitetura de controlo, dentro do mesmo cenário de aplicação, considerando o caso em que o alvo, após a sua deteção durante o modo de busca, deverá ser seguido de forma colaborativa. A implementação do modo de seguimento teve como foco o desenvolvimento de um controlador que permite a um conjunto de aeronaves seguir um caminho circular, centrado em torno de um alvo e que se move solidário com este, usando o método de *Moving Path Following* (MPF) (Oliveira, Aguiar e Encarnação (2016) e Jain (2019)).

Numa primeira fase, a arquitetura de controlo proposta é validada através de simulações numéricas, considerando um modelo do tipo uniciclo para as aeronaves e alvo simulado. Em seguida, adota-se uma abordagem mais realista, usando simulações com *software* no *loop*, considerando um modelo de fonte aberta para a dinâmica das aeronaves consideradas. Nesta fase são desenvolvidos módulos de *software* específicos para o sistema de controlo considerado nesta tese, usando uma arquitetura com base em Python, ROS, PX4 e Gazebo. Finalmente, depois de implementadas e validadas estas simulações, uma nova capacidade é acrescentada: um algoritmo de deteção de alvos com base em visão computacional para estimar dados sobre o alvo, baseado nas imagens obtidas por uma das câmaras montadas numa das aeronaves não tripuladas, usando o módulo de deteção automática desenvolvido por Alves, Oliveira, Cruz e Silva (2022). Esta simulação permite fechar o sistema de controlo implementado usando um sistema de visão computacional.

Os resultados obtidos demonstram a eficácia do método proposto, num contexto de simulação computacional realista, que permite uma transição expedita para os ensaios em voo.

Palavras-chave: Controlo não linear, Coordenação de voo, Seguimento de caminho em movimento, UAVs

Abstract

The importance of Unmanned Aerial Vehicles (UAV) applications is becoming increasingly clear with every new year. While a lot of research has been done in the past, there are still several research areas, topics and applications to unveil and explore. The Air Force Academy Research Center (CIAFA) is working on a new modular control architecture for UAVs that allows for the reutilization of several modules such as trajectory control, computer vision, automatic learning and sensory data fusion, applied to a large spectrum of missions, which includes automatic landing or terrestrial target tracking based on computer vision.

This thesis contributes to the formal implementation of a command and control architecture, to be used in search and target tracking missions in a maritime environment. The architecture was idealized in Santos (2021), using three main operating modes: search mode, target tracking mode and collision avoidance mode. Specifically, this dissertation contributes to the implementation of the target tracking mode of that architecture, in the same application scenario, considering that after detecting the target, this should be followed in a collaborative way. The main focus of this target tracking mode implementation is a controller that allows for a group of aircrafts to follow a circular path centered at the target coordinates and moves together with it, using the Moving Path Following (MPF) method (Oliveira, Aguiar, and Encarnação (2016) and Jain (2019)).

In a first phase, the proposed control architecture is validated through numerical simulations, considering kinematic unicycle type model for the aircrafts and a simulated target. Next, a more realistic approach is adopted, with software in the loop simulations, using an open source model for the considered aircrafts dynamics. In this phase, specific software modules are developed for the considered control system, using an architecture based on Python, ROS, PX4 and Gazebo. Finally, after the implementation and validation of this simulations, a new feature is added: a target detection algorithm based on the received data from one of the UAVs onboard cameras to estimate the target's position and velocity, using an automatic detection module developed in Alves et al. (2022). This simulation closes the implemented multi-UAV control loop system using a computer vision algorithm.

The obtained results show the efficiency of the proposed method in a realistic computational simulation context, which allows for a quick transition to eventual flight tests.

Keywords: Flight coordination, Moving Path Following, Non-linear control, UAVs

Contents

List of Figures

List of Tables

Abbreviations

CIAFA Air Force Academy Research Center

FAP Portuguese Air Force

LQR Linear-Quadratic Regulator

MPF Moving Path Following

PI Proportional-Integral

PID Proportional-Integral-Derivative

QGC QGroundControl

ROS Robot Operating System

UAV Unmanned Aerial Vehicle

UAVs Unmanned Aerial Vehicles

Nomenclature

Greek symbols

ϕ	Roll Angle.
ψ	Yaw angle.
θ	Pitch angle.

Roman symbols

T	Thrust.

Subscripts

d	Particle.
r	Robot.
t	Target.

Superscripts

’	Transpose.
I	Inertial frame.
R	Robot frame.
T	Target frame.

Chapter 1

Introduction

1.1 Motivation

Up until the year of 2022, the Portuguese Air Force (FAP), more specifically, the Air Force Academy Research Center (CIAFA), has been involved in several important research projects such as the PITVANT, PERSEUS, SUNNY, SEAGULL or FIREFRONT, all using unmanned aircrafts (see AFA (2022)). These mainly focused on the development of tools and added capabilities and in the designing, manufacturing and operation of Unmanned Aerial Vehicles (UAV). Among the stated projects, the main publications include target tracking and path following for fixed wing UAVs (Oliveira, Aguiar, and Encarnação (2016) and Oliveira, Aguiar, and Encarnacao (2017)), computer vision (Marques, Bernardino, Cruz, and Bento (2014a), Marques, Bernardino, Cruz, and Bento (2014b) and Cruz and Bernardino (2015)), multidisciplinary design optimization (Felix, Gomes, and Suleman (2013)), micro and nano flapping-wing aerial vehicles (Armanini, Caetano, de Croon, de Visser, and Mulder (2016)) and reliability and airworthiness for small UAVs (Gonçalves, Sobral, and Ferreira (2016) and Gonçalves, Sobral, and Ferreira (2017)). The creation of Squadron 991 in November 2021 for unmanned aircrafts shows the growing interest and investment of FAP in this type of technology. "Harpias", the most recent FAP squadron, currently operating the UAVISION OGASSA OGS 42N/VN, is designated with missions like maritime and terrestrial surveillance, support to firefighting related activities and surveillance and reconnaissance (FAP (2022)). The scientific efforts in this area of research are of extreme relevance, given the utility and advantages this type of aircraft adds in the accomplishment of the Portuguese Air Force attributed missions.

Figure 1.1: UAVISION OGASSA OGS 42N/VN, currently used by the Portuguese Air Force Squadron 991 "Harpias" (FAP (2022)).

The coordination of a team of autonomous vehicles allows to accomplish missions that no individual au-

tonomous vehicles can accomplish on its own. There are several advantages of using multiple UAVs when compared to single powerful Unmanned Aerial Vehicle (UAV) usage. According to Maza et al. (2015), using more than one UAV allows multiple simultaneous interventions, meaning that these machines can perform in different spatial locations at the same time, whether to get information from different points in space or to apply forces on a load from different angles or positions, for instance. It also achieves greater efficiency, since mission time can be decreased in some specific tasks like searching for targets or exploring large areas. Having more UAVs also increases the reliability of the operation, offering redundant solutions with greater fault tolerance and flexibility (even if one of the machines has a problem like a faulty sensor, others may replace its role and different configurations can be used). Regarding the costs, having a lot of low-cost UAVs could be cheaper than having a single expensive solution in cases where constraints like power consumption, weight and size play an important role (Hamilton and Ochmanek (2020)). A team of UAVs supports the idea of complementary assistance in which different roles can be assigned specifically to each member; as an example suggested in Maza et al. (2015), one member can be a fixed-wing airplane that usually has a longer flight range and endurance and the other can be an helicopter with vertical take-off and landing capabilities.

It is intended, with this thesis, to continue the work done in Santos (2021) and Alves et al. (2022), where a command and control system for multiple vehicles to work on maritime environments is developed. Specifically, Alves et al. (2022) implemented a new command and control architecture for low cost UAVs, using a hardware and software framework which includes, among others: Autopilot PX4 (PX4 (2022)), Robot Operating System (ROS)/MAVROS (ROS (2022)) and Open CV (OpenCV (2022)). This framework was developed in a modular way, using as a case study the detection of a target (ground vehicle) and respective automatic following based on computer vision techniques and machine learning. As a sequence to that work, the formal implementation of a command and control architecture to be used in target search and following in maritime environment was idealized in Santos (2021) using three main modes: search mode, following mode and collision avoidance mode. More specifically, Santos (2021) considered the application on UAV's flight formations of search patterns based in NATO publications (International Civil Aviation Organization (2016)), considering the use of different onboard sensors (different spectrum cameras) distributed through the vehicles. In that work, classic path following and trajectory tracking controllers were also implemented as a solution to the flight formation control for the search mode in maritime environment missions.

Finally, this thesis proposes the implementation of the target tracking mode of the control architecture, in the same mission scenarios, considering the case where the target, after being detected, should be followed in a collaborative way.

1.2 Thesis Outline

This dissertation is divided into six main chapters, each with its own sections and subsections. In the first chapter, the motivation for the development of this work is addressed, as well as the contributions to the scientific community and the problem formulation, which specifies the questions to be addressed. The second chapter lays the bibliographical foundation to answer the proposed questions in the problem formulation; several articles and implementations are discussed, as well as the theoretical background regarding different ways of implementing the

solutions. Afterwards, in the third chapter, the chosen solutions are implemented and validated through numerical simulations. Then, in a subsequent validation stage, these kinematic solutions are implemented together with a UAV dynamic model resorting to an open-source software in the loop tool, thus allowing for the validation of the proposed control system in a realistic setup. In the fourth chapter, the control loop is closed using a computer vision algorithm to estimate the target position based on acquired images from an electro-optic sensor onboard of one of the UAVs. The last and sixth chapter concludes the dissertation, discussing the results obtained, future work to be developed and summarizing the overall project.

1.3 Contributions

With the project developed being integrated in the works of CIAFA, it is important to recognize what contributions were given with the results of this dissertation. Since 2020, CIAFA started developing a new command and control architecture for target detection and following both in land (Alves et al. (2022)) and maritime Santos (2021) environments.

This architecture follows a modular approach which includes a high-level decision algorithm that switches between different flight modes such as collision avoidance, formation flight and collaborative tracking. In a lower-level, several different applications can be included such as trajectory control, computer vision, communications, filtering and data fusion.

In this context, this dissertation contains three main important contributions:

1. Development of a computational architecture to command and control multiple vehicles;
2. Implementation of a motion control strategy that allows a set of multiple vehicles to follow, in a distributed architecture, a moving target.
3. Improvement of CIAFA's control software architecture, allowing for the use of the most updated tools for simulation.

1.4 Problem Formulation

With the motivation unveiled, it is now time to give a general overview and to formulate the problem to be solved with the work of this thesis.

Following the work done in Santos (2021), in which an architecture was defined according to the scheme in Figure 1.2 and the search mode was implemented, this thesis proposes the implementation of a target tracking mode that was not implemented previously.

In this thesis, it is intended to implement a control system that allows a group of N UAVs to follow in a coordinated way a target in a given environment, using a computer vision system to estimate the target's position. Regarding the use of multiple vehicles, one must answer the following questions:

1. Which geometry is the most suitable to solve the problem in the target tracking phase?
2. Which control architecture should be implemented?

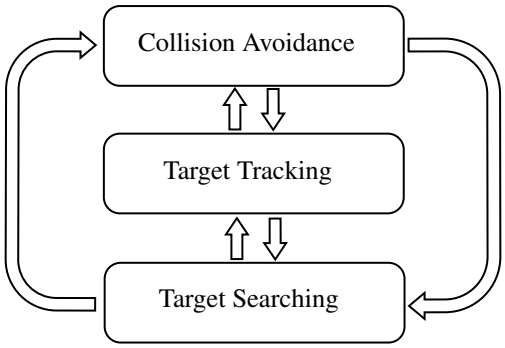

Figure 1.2: Control hierarchy defined in Santos (2021).

3. Which motion control technique is the most suitable?

The next chapter explores some possible ways to implement the solutions, in order to answer the posed questions.

Chapter 2

Literature Review

In this chapter, important concepts are explored and defined to serve as a basis for the implementation work developed in the next chapters. It is divided in four main topics, each with important relevance to this thesis work and subdivided into more specific theoretical concepts. These topics are Multi-UAV Physical Architectures in Section 2.1, Multi-UAV Control System Architectures in Section 2.2, Vehicle Formation in Section 2.3 and Motion Control Techniques in Section 2.4.

2.1 Multi-UAV Physical Architectures

Based on Maza et al. (2015) one possible scheme of classification for multi-UAV architectures is defined according to the coupling between the members. According to this classification, there are four types of couplings: physical coupling, formations, swarms and intentional cooperation.

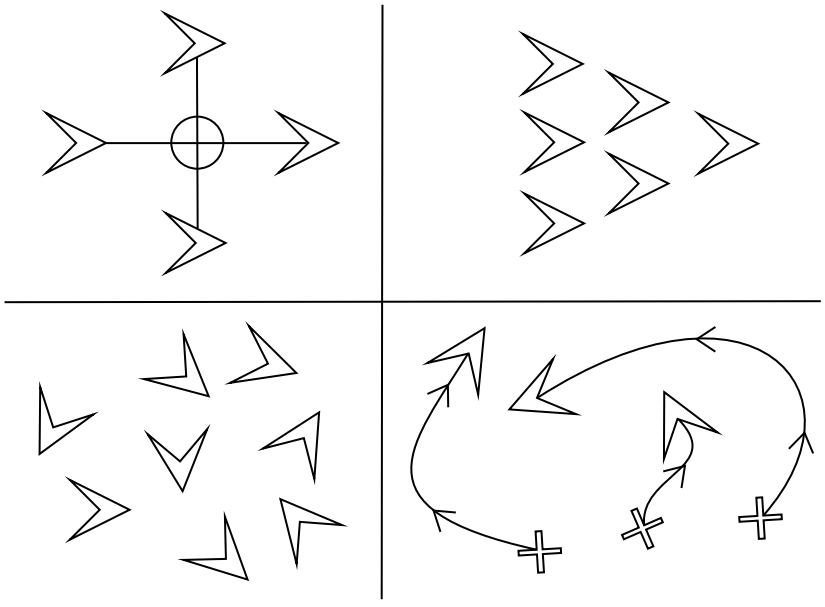

Figure 2.1: Illustrative scheme of multi-UAV architectures classifications, based on Maza et al. (2015). From left to right and top to bottom: physical coupling, formation, swarm and intentional cooperation.

As the name implies, physical coupling is based on a physical connection between each member of the team

which directly constrains their motions depending on the forces applied by and on each other. This kind of system is used traditionally to lift and carry heavy loads, that one single UAV would not be able to. Using this scheme implies a more complex and complicated physical system that is usually harder to model and control. In terms of collision avoidance and motion planning, the entire team is considered as a whole. Some practical applications in the literature can be found in Maza, Kondak, Bernard, and Ollero (2009) and in Bernard, Kondak, Maza, and Ollero (2011).

The second classification is called formation. In this architecture, although there are no physical constraints applied, the motions of all members relative to each others are heavily constrained so they can follow a specific shape in a coordinated manner. By using this strategy, one can easily introduce new members either to extend or to replace faulty ones; it is also a very stable approach, strongly researched with proposed robust controllers that allow for insensitivity to uncertainties in the motion of nearby agents and also to delays in the transmission of information in the feedback path. As in the first architecture proposed, for motion planning the formation is considered as a whole. But, regarding the collision avoidance problem within the team, this can also be included in the formation control strategy. In the literature, Yun, Chen, Lum, and Lee (2010) and Paul, Krogstad, and Gravdahl (2008) use this architecture, with leader-follower and virtual leader approaches, respectively.

Another type of architecture referenced in Maza et al. (2015) is the swarm type. These consist in a homogeneous team of many vehicles in which the resulting motion does not necessarily lead to formations, but the interactions between the members lead to emerging collective behaviours. The main advantage is having a lot of "unintelligent" agents with simple programmed interactions develop complex collective global behaviours. According to Abdelkader, Güler, Jaleel, and Shamma (2021), recent research efforts show scalability issues of current swarm systems in real world environment, since there are not a lot of works reporting outdoor experiments with small number of UAVs, mainly due to limited confidence on the swarm-level relative location, the rate of exchanged information between members and even the individual robot's location. Throughout the research published, one can find applications of swarms architectures in different areas such as in entertainment, security and surveillance (Nigam, Bieniawski, Kroo, and Vian (2012) and Petrlik, Vonasek, and Saska (2019)) or exploration (Rosalie et al. (2017)).

The final classification is intentional cooperation. In this architecture, each member has individual tasks assigned (like a predetermined path) and they cooperate to perform a global mission in an optimal planned way (Parker (1998)). Opposed to the formation scheme discussed before, there is no need to have a geometric relation between them. It also allows for a team of vehicles to collect information from multiple locations and exploit this information to build models that can be used to make decisions across individual or small groups; the knowledge extracted from each UAV is merged and global cooperative perception is performed. Depending on the mission to be executed and the robots used, several extra problems should be solved like multi-UAV task allocation (which UAV does which task), high-level planning, plan decomposition, and conflict resolution (collision avoidance and crossing trajectories) (Parker (1998)). This classification can be attributed in Gancet, Hattenberger, Alami, and Lacroix (2005) and Viguria, Maza, and Ollero (2010), where coordination between aerial and ground vehicles is achieved and is addressed more thoroughly in Han, hong Wang, and xing Yi (2013) and in Goerzen, Kong, and Mettler (2009).

2.2 Multi-UAV Control System Architectures

According to Ridao, Yuh, Batlle, and Sugihara (2000), control architecture is defined as the framework that manages both the sensors and actuators systems and thus enables the robot to undertake a user specified mission. In this section, the way the controllers and command centers are connected to the UAVs is explored in further detail. Following Peng et al. (2021), the control architecture can be divided into three main classifications, that are chosen depending on the available communication bandwidths and sensing abilities. These architectures are centralized control, decentralized control and distributed control.

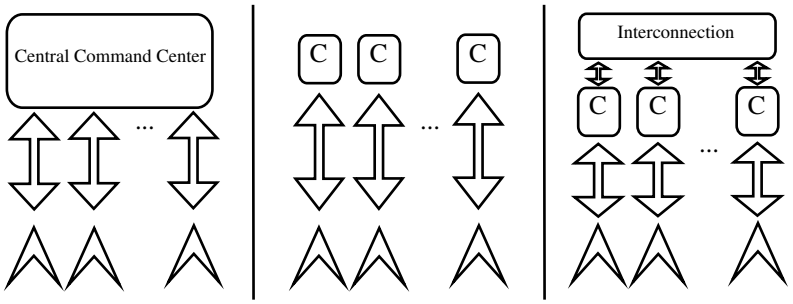

Figure 2.2: Illustrative scheme of UAV control architectures, based on Peng et al. (2021). From left to right: centralized control, decentralized control and distributed control.

Starting with centralized control, this can be defined as a scheme in which there is a central command center that acquires all the robots information, processes the information, makes the decisions and then sends the calculated control inputs to each UAV. The command center can be either a ground station or a designated UAV and all the information gathers in that single point; this leads to a global optima when optimizing for the objective, instead of a local optima like in the case of decentralized control. Another positive aspect is the fact that conflicting actions could be resolved and avoided among the vehicles. The negative points are the considerable computational power and communication bandwidth required when the number of UAV grows. Also, if there is a problem with the central command center (for instance, a leader UAV crashes), this may lead the system to an unstable behaviour. Examples of applications of this architecture can be found in Arrichiello, Chiaverini, and Fossen (2006) and Skjetne, Moi, and Fossen (2002). This architecture is usually applied both to physical coupling and to intentional cooperation (Maza et al. (2015)).

For decentralized control, the UAVs are connected individually and exclusively to one own command center. The information about each vehicle is only available to itself, which leads to optimization following the local optima obtained for each member of the team. The main advantages are related with the properties of modularity and scalability and also the fact that it can tolerate individual failure. Simply put, it is similar to the design methodologies followed by single vehicle control systems. For reference, this architecture can be found in Liu, Wang, Peng, Chen, and Li (2019) and Dai, He, Lin, and Wang (2018). This architecture is mandatory for swarms, preferred for formations and it can be used in both physical coupling and intentional cooperation (Maza et al. (2015)).

Lastly, the distributed control architecture consists in having a local control law implemented for each UAV but with exchange of information between the control laws through local sensing and communication. It is a far more complex architecture in structure and organization but it does not require a central command station so

it is more fault tolerant. Compared to the other two, this scheme is more promising when considering limited computing resources, short communication ranges, narrow communications bandwidths and large size of vehicles to maneuver and control (Murray (2007)). It is also important to add that graph theory plays an important role when designing a distributed control architecture for multi vehicle systems (Murray (2007)). In the literature, some example implementations can be found in Peng, Wang, and Wang (2017) and Peng, Wang, Li, and Han (2020).

2.3 Vehicle Formation

It is now time to formulate the different possible vehicle formations and how these machines can interact with each other to accomplish a certain specified goal. In the literature, the main types are consensus based formations (which divides itself in the classical leader-follower, virtual structure and behavioral approaches, according to Ren (2006)), artificial potential functions and graph-based methods. All these possible formations will be discussed in more detail in this section.

2.3.1 Consensus Based Formations

According to Ren (2006), there are three different vehicle formations inside the group of consensus based formations: leader-follower, virtual structure and behavioral. All this approaches can be derived as special cases from a general case, defined in the paper. For each one of them, a controller is defined based on their defining base characteristics, explored further in this subsections.

Starting with leader-follower, this approach selects one leader vehicle to be the reference for the remaining follower vehicles. As P. Wang and Hadaegh (1996) describe it, the leader serves as the "skeleton pattern" for the fleet and the desired motion is determined with respect to what the position of the leader might be. It is considered a simple approach according to Hejase, Noura, and Drak (2015) since, to dictate the groups behaviour, it is only necessary to specify the leader's motion. The main disadvantage is having a high dependency on the leader and if this machine fails, the whole formation is compromised. Throughout the literature, in Fahimi (2007) and in Shojaei (2015) one can find this method applied. Santos (2021) also used this method for his control architecture implementation on a target search mission in maritime environments.

In a virtual structure approach, the principle is similar to the one described in the paragraph before: one virtual vehicle or point is created and serves as the leader/reference to all the UAVs in the formation. It has the same advantages named before and also the extra one of not collapsing the formation if one of the UAVs fails. The applications of this method are limited, due to the its rigid implementation (Fahimi (2007)). An example of a practical implementation in the literature can be found in Skjetne et al. (2002).

Finally, in the behavioral approach the main formation control task is broken down into smaller tasks that are usually referred as behaviours (Hejase et al. (2015)). After broken down, each task is assigned with a weight based on the prioritization level and then these values are used to define the plan of actions during the mission execution. Due to this unique dynamics, the system behaviour and movements might be difficult to predict but it makes it easier to prioritize and account for the different tasks. In Arrichiello et al. (2006) this method is used to control marine surface vessels.

2.3.2 Artificial Potential Functions

In the artificial potential functions approach, Khatib (1985) states that the philosophy is described as having attractive poles for the position to be reached and repulsive surfaces for obstacles to be avoided. In Leonard and Fiorelli (2001), this method is applied inspired in biological models design by scientists in this field, suggesting three main elements to maintain a group structure: attraction to distant neighbors up to a maximum distance, repulsion from neighbors too close and alignment or velocity matching with neighbors. Practical applications of this approach can be found in Tazibt, Achir, Muhlethaler, and Djamah (2018) and more recently in Souza et al. (2022).

2.3.3 Graph-Based Methods

This methods are based on graph-theory in which each vehicle is modeled as a node and the connections and interactions between them are represented as connecting networks. It allows for a mathematical representation of the vehicle formation and serves as a basis for the communications between each element. Examples in the literature can be found in Peng, Wang, and Wang (2018) and Peng et al. (2017).

According to Peng et al. (2021), in practical terms all this methods can be thought as classic motion control problems, such as path following and trajectory tracking. In path following problems, one intends to design a controller that makes the vehicle converge to a determined path with a determined velocity. In trajectory tracking the objective is similar but the path and velocities desired are time dependent. In Figure 2.3 is possible to perceive the difference in a graphical way: while in the first case the vehicle starts to converge to the path and as time passes it adjusts its convergence, in the second, the vehicle is always trying to converge to predetermined points along the path, independent of time.

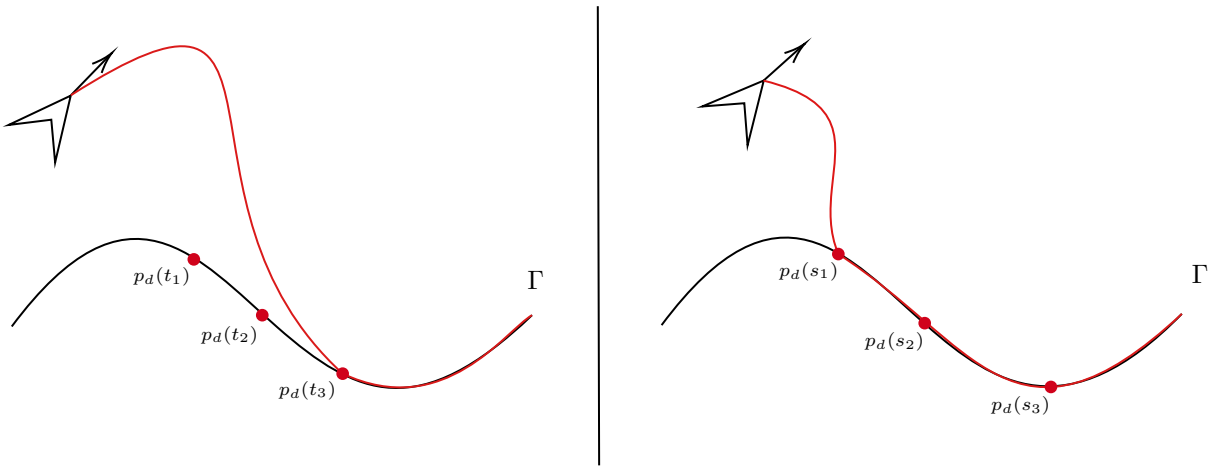

Figure 2.3: Illustrative comparison between path following (on the right) and trajectory tracking (on the left), based on Brahim (2018).

There is a third type of classical motion control problem referred by Peng et al. (2021) called target tracking in which the main objective is to drive the vehicle to track the position of a moving target with a relative range and

orientation. The target trajectory is usually not known in advance and only its instantaneous motion is available, varying through time. In these cases, the data on the target should be obtained in a collaborative effort by multiple vehicles, since the sensors available for each individual vehicle might not be enough to do it autonomously and independently. Being so, the vehicles should work in a collaborative way to obtain this data, when there are robust and sufficient communication channels between the elements. Examples in the literature can be found in Breivik, Hovstein, and Fossen (2008) and Shojaei (2016).

2.4 Motion Control Techniques

To solve the problems stated above, different control techniques can be used. In this section, Proportional-Integral-Derivative (PID), Linear-Quadratic Regulator (LQR), gain scheduling, adaptive control, sliding mode control, model predictive control, backstepping and Lyapunov vector fields techniques are explored in further detail. Although not detailed in this thesis, it is also important to refer that there are other important used techniques in the literature like dynamic programming and piecewise affine control.

PID controllers, as the name suggests, are composed by three main sub-controllers: a proportional, a derivative and an integrative. After calculating the error, defined by the displacement between the reference value (input) and the current system state (output), this value goes through this different parts and in the end gets added together to serve as a control input to the system (see Figure 2.4). Important parameters are the gains for each sub-controller (usually denoted by the letters k_p, k_i and k_d) that should be tuned to increase the overall performance of the controller.

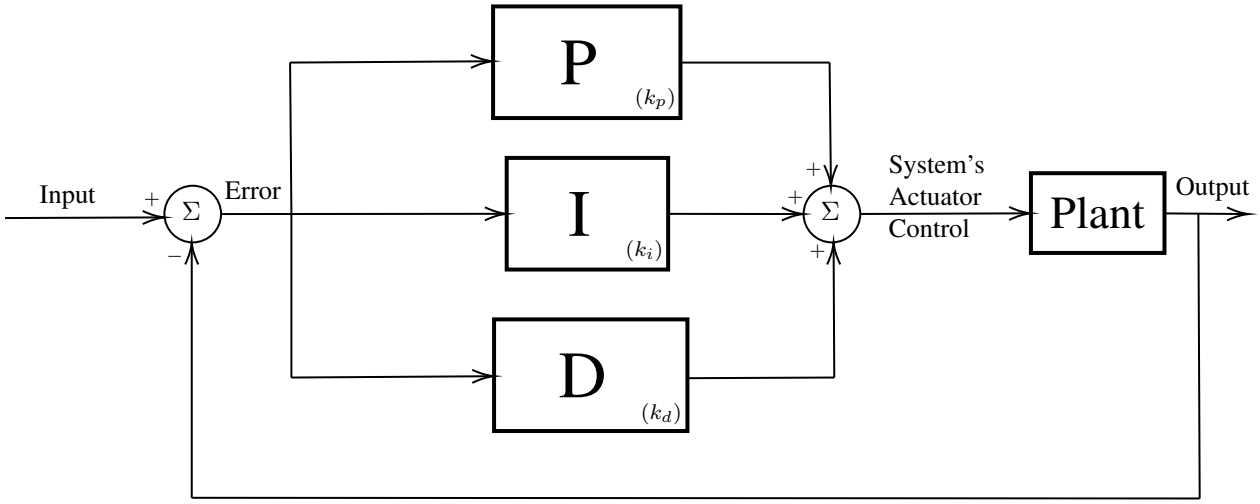

Figure 2.4: Representative scheme of PID controller.

For LQR controllers, the system needs to be either linear or linearized. It provides a solution for an infinite-horizon, continuous-time linear quadratic problem in an automated way (B. Wang, Dong, and Chen (2010)) by minimizing a specific cost function. The control input solution to minimize this function can be obtained by solving the algebraic Riccati equation and using it to formulate the control input. Illustrative examples of application of LQR controllers can be found in B. Wang et al. (2010), in Ratnoo, Sujit, and Kothari (2011) and in Lee, Cho, and Kee (2010).

Following the ideas in Leith and Leithead (2000), gain scheduling was once one of the most popular control techniques due to its "divide and conquer" approach: taking a non-linear problem, if possible, then one can divide it into subproblems and apply already well studied and established linear design control methods onto each of this divisions, like a PID (Milhim, Zhang, and Rabbath (2010)). This decomposition into linear subtasks can be done, for instance, through series expansion linearization about a single trajectory or equilibrium point. In the literature, it is possible to find this method applied in Cunha, Silvestre, and Pascoal (2003).

According to Tao (2014), adaptive control is a control methodology that allows the incorporation of variables related to system parametric, structural, and environmental uncertainties caused by internal or external disturbances and variations. As these variables change, different control parameters are used to adjust to the newly generated scenario. It is a methodology that involves a large number of subfields of study like adaptive predictive control, adaptive learning control, stochastic adaptive control, etc. Given a system represented by its space state dynamics with unknown parameters, this methodology allows to achieve system stability (signal boundedness) and asymptotic tracking, despite system parameter uncertainties. Based on the performance errors, the controller's parameters are changed automatically or adaptively to reach the goals proposed in the previous sentence. Application examples of this technique can be found in Kaminer et al. (2007) and in Cao, Hovakimyan, Patel, Kaminer, and Dobrokhodov (2007).

In Healey and Lienard (1993), the idea behind sliding mode control is constructing sliding surfaces, that are defined in the state error space, to find a sufficient relationship for each of the control element inputs that guarantees global stability of the state variable errors and provide adequate performance under closed loop conditions. The final obtained control input will depend on the sum of each of the derived control inputs obtained for each occasion (in the paper, three different ones for the underwater vehicle: steering, diving and speed), and each of this depends on the sliding surfaces which are assumed to be known. In Nelson, Barber, McLain, and Beard (2007), this technique was used for controlling miniature air vehicles.

Model predictive control is an optimal control technique that finds the optimal control sequence in discrete time to minimize a specific cost function, being subjected to some set of restrictions on the state variables or control input. The parameters that compose the cost function should all be properly selected (Li, Sun, and Oh (2010)). It is applied in the previously referenced paper for marine surface vessels and in Mansouri, Nikolakopoulos, and Gustafsson (2015) for UAVs.

Based on the explanations in Raptis and Valavanis (2011), the backstepping control technique takes a system defined by its state variables and respective dynamics and, with the ultimate objective of defining a control law that takes those states to zero as time goes to infinity, recursively rewrites them to depend on the control input. This technique is applied in UAV applications in Chitrakaran, Dawson, Kannan, and Feemster (2006) and Ahmed and Subbarao (2010).

As the name suggests, Lyapunov field vectors is a control technique in which a grid of vectors is generated in 2D or 3D space and depending on the UAV location on that grid, a different vector is considered so that the vehicle can follow or reach the predetermined path. A Lyapunov function takes an equilibrium point at a minimum value and decreases along solution curves in a neighborhood of the equilibrium point (Suda (2019)), creating a smooth function; an example of a Lyapunov function is the potential function of gradient vector fields. This technique is typically used to track loiters, producing globally stable results Sujit, Saripalli, and Sousa (2014). To construct this

fields, Suda (2019) proposes methods based on linear programming when working with linear systems; other more general methods can be used and are referred with more detail in this article. In Frew, Lawrence, Dixon, Elston, and Pisano (2007), results of the application of Lyapunov field vectors in a micro UAV are shown.

Chapter 3

Flight Control System's Implementation

3.1 Considered and Proposed Solutions

Given the literature review done in Chapter 2, the next sections aim to find the answers to the questions stated in the problem formulation introduced in Chapter 1. Initially, several solutions are considered and subsequently, the actual chosen solutions are explained in a justified manner at the end of each subsection.

3.1.1 Moving Path Following Controller

The first path following and trajectory tracking applications were originally introduced by Samson et al. (see for example de Wit, Khennouf, Samson, and Sordalen (1994) and its references). These concepts were then applied and implemented in different types of vehicles, for instance in Aguiar and Hespanha (2007), Kaminer et al. (2010) and in Encarnacao and Pascoal (2000). To solve the ambiguity on the path following formulation, Soetanto, Lapierre, and Pascoal (2003) proposed the concept of a virtual particle point, using the vehicle's velocity command as an additional degree freedom.

The concept of Moving Path Following (MPF) is originally described and developed in Oliveira, Aguiar, and Encarnacao (2016). The MPF method is applied to the general case of desired paths moving with respect to an inertial coordinate frame with time-varying linear and angular velocities, and with non-constant curvature and torsion. Thus, the MPF method generalizes the classical path following for stationary paths, thus providing a generic tool to follow moving (time-varying) paths that can be applied to several mission scenarios, like target tracking, thermals soaring or gas clouds monitoring. In Oliveira, Aguiar, and Encarnacao (2016), after deriving the MPF error kinematic model, a generic control law is derived for planar paths. The solutions obtained were implemented using numerical simulations and then, flight test experiments were conducted with a single fixed wing UAV (ANTEX-X02), both for single and multiple targets. The results obtained showed the effectiveness of the proposed method. Finally, Jain (2019) considered the already formulated MPF problem and extended it to multiple vehicle applications.

Proposed Solution

By design, the MPF method retains all the desirable characteristics of the classical path following method, namely smooth convergence to the moving path and the possibility of doing so at constant speed with respect to an inertial coordinate frame (Oliveira, Aguiar, and Encarnacao (2016)). Additionally, the MPF was successfully tested and validated in practical applications with CIAFA's UAVs.

In this thesis, the controller that computes the inputs to steer the vehicle to the virtual particle point along the circular path is based on the work developed in Jain (2019). All the details regarding this Lyapunov-based controller, including the corresponding error dynamics derivation and proof of convergence are presented. Other examples of successful MPF implementations can be found in Guan, Liu, Zheng, Ma, and Zhu (2022) and Rahmaniar and Santoso (2022).

The coordination system to get the vehicles equally spaced in the circle is performed by controlling the progression rate of the particle in the circle and having a leader and virtual particles to be followed, as done in Y. Wang, Wang, and Zhu (2019a). These particles are controlled to be equally spaced and to move at a desired speed, being the control architecture applied separately from the vehicle steering control, in a chain like network.

3.1.2 Vehicle Formation

Several types of formations were explored by analysing already implemented published articles. An example application for circular formations in multiple vehicles missions can be found in Zhang and Liu (2015). With the work of Briñón-Arranz, Seuret, and Pascoal (2017) and Briñón-Arranz, Seuret, and Pascoal (2019), target tracking controllers via a circular formation for unicycle-type vehicles are described, but the latter describes one where the access to information is limited. To achieve this, it only uses the velocity of the target and the relative positions of the agents with respect to it, expressed in the local frame of each vehicle. An "exosystem" is implemented to generate the circle trajectory to follow around the target. The spacing between vehicles is maintained through communication between the agents. Throughout Briñón-Arranz et al. (2017), several other papers implementing circular formations are mentioned. In Yu and Liu (2016), a controller that guarantees that the vehicles describe a circular trajectory around a common fixed center using only local information and cyclic pursuit techniques was implemented, for instance.

Swarm implementations for target tracking can be found in Brust et al. (2017), where the authors explore natural biological phenomena to implement what they call as Based on Dual-Pheromone Clustering and more recently in Zhou, Liu, Li, Xu, and Shen (2021) where deep reinforcement learning is used.

Proposed Solution

In order to follow the target, a circular path around it with a predefined constant radius is created so that the UAVs can follow it. The main advantages of this choice are: it is easy and simple to mathematically design and define a circular path, with an adjustable radius; it is applicable to any number of vehicles (considering a minimum safe distance); fixed wing airborne vehicles can be always moving along this path, meaning that even when the target is steady, the aircraft still has a path to follow, never having the velocity going to zero. Furthermore, the UAVs are chosen to be equally spaced inside the circle, which ensures a maximum distance between vehicles and

helps with a safer operation in terms of collision avoidance, to be implemented in future works.

3.1.3 Control Architecture

Detailed examples of implementation in the literature for each type of architecture were taken into account. Regarding centralised architecture, Bella, Belbachir, and Belalem (2019) use an unmanned aircraft to control, monitor and survey a set of unmanned surface vessels with the main purpose of collecting sea garbage. A simple centralised architecture is introduced with several hierarchical categories, depending on the vehicles type and mission.

The work done in Jain (2019) explores decentralized architectures for multiple mobile robotic vehicles. It develops a framework that is composed by two subsystems, the MPF and the "Dynamic Event Triggered Cooperative Control". The main advantage of this application lies in the flexibility to have different formations and maneuvers, opposed to other methods generally used in which they are specifically designed for circular patterns. It is also mentioned in this thesis the relevance of the communication system in the coordinated control of the vehicles, in which a possible solution to avoid constant communication between the team is to use event-based sampling techniques like event-triggered control approaches. Lastly, this work explores strategies to solve the localization source problem.

In Elston, Frew, and Member (2008), for instance, a distributed architecture is presented. The main idea is to use central aircrafts (coined as the "motherships") with specific characteristics such as endurance, range, and processing capabilities and then have them act as interconnections for smaller, stealthier, more flexible and maneuverable swarms of "daughterships". There is added complexity due to the definitions on communication protocols since there are a several different actors with very specific patterns of connections between them.

Proposed Solution

Given the considered mission scenario in this thesis, where the UAVs should be flying relatively close to each other, it assumed that a communication network connecting all the participating UAVs is available. Having shared data on the target between UAVs to obtain the best possible estimations, possibly obtained by image acquisition and data fusion algorithms, can lead to better estimates and better mission performances.

A distributed architecture is chosen and it is assumed that each vehicle knows the information about the vehicle that it is following, creating a chain communication network. The information shared between the vehicles is the rate of progression of the virtual particle and its location. The chained network starts at a virtual leader particle and ends at the last N UAV (Y. Wang, Wang, and Zhu 2019b). This allows for reduced communication since there is no need for every vehicle to know the information of every other vehicle and only the information of the vehicle that it is following is required.

This thesis focus on the implementation of a MPF controller that allows a group of UAVs to track a moving ground target, in a distributed way. Being so, it is assumed that the data on the target's position and velocity should be acquired from a external system.

The data acquisition system is not discussed in this thesis. Nevertheless, at a final stage of implementation in Chapter 5, a computer vision system onboard of one of the UAVs is considered (based on the work developed in

Alves et al. 2022), which sends the target's estimations to the all the remaining UAVs in the flight formation, in a centralized manner.

In the next section, the equations that compute the control inputs and coordinate the particles positions along time are shown with every term explained explicitly.

3.2 Mathematical Model and Control Laws

To implement the MPF controller, solve the problem stated previously and validate the results obtained, the software MATLAB (MathWorks (2022a)) was used. The controller can be divided into two main core sections: the linear and angular velocity controller to steer the vehicle to follow the desired moving path and the particles controller, that specifies the position of a virtual particle on the desired path for each UAV, ensuring that they are equally spaced and moving at the desired speed. The following subsections show the equations that define this controllers and the respective proofs of convergence.

3.2.1 Moving Path Following Controller Using Virtual Particles

In Jain (2019), a controller for the linear and angular velocity is derived and implemented. The controller computes the commands by taking into account a given generated virtual particle, following it with an error ϵ arbitrarily small, as explained further. In this section the equations that compute the control inputs are shown and explicitly derived and convergence is proven through Lyapunov-based methods.

3.2.1.1 Problem Definition and Notation

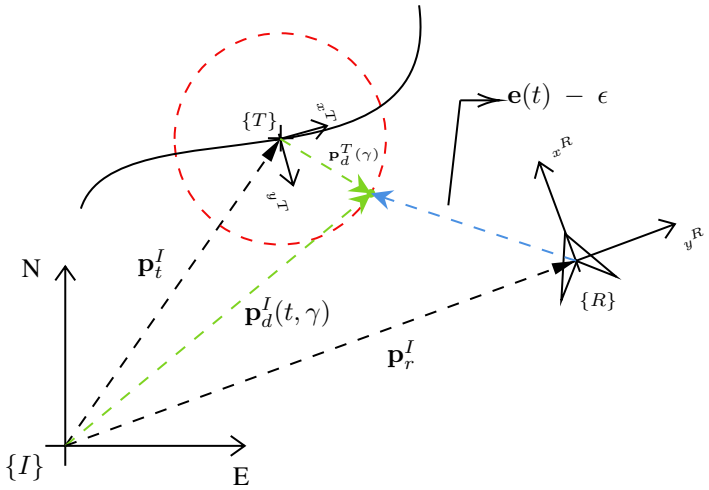

Figure 3.1: Coordinate frames, according to Jain (2019).

Given, in two dimensions (see Figure 3.1), an inertial reference frame $\{I\}$ and a target frame $\{T\}$ attached to a moving target with unknown dynamics, let the position and velocity of this target be defined as $\mathbf{p}_t^I \in \Re^2$ and $\mathbf{v}_t^I \in \Re^2$, with respect to the inertial frame, respectively; it is assumed that in every time instant, these two variables are known. Let $\mathbf{p}_d^T : \Re \to \Re^2$ be a defined fixed reference geometric path parameterized by $\gamma \in \Re$, with $\dot{\gamma} \in \Re$ being the desired speed assignment. The parameter γ represents, for instance, the arc length inside the path and

it can be seen as a virtual particle along reference path. Being so, the speed of the virtual particle, $\dot{\gamma}$, dictates the evolution of this virtual particle along the path over time.

The vehicle kinematics are given by equations (3.1).

$$\begin{aligned} \dot{\mathbf{p}}_r^I(t) &= R_R^I(t)\mathbf{v}_r^R(t) \\ \dot{R_R^I}(t) &= R_R^I(t)S(\omega_r) \end{aligned} \tag{3.1}$$

$\mathbf{p}_r^I \in \Re^2$ is the position of the vehicle expressed in the inertial frame, $\mathbf{v}_r^R = [v_f(t)\ 0]'$ is the linear velocity of the vehicle expressed in the robot body frame $\{R\}$ and $\omega_r(t)$ is the angular velocity of the vehicle. The two dimensional rotation matrix $R_R^I(t)$ represents the orientation of the vehicle frame $\{R\}$ with respect to the inertial frame of reference. The control input is defined as $\mathbf{u}(t) = [v_f\ \omega_r]'$ and is constrained, meaning that $v_{min} < v_f < v_{max}$ and $\omega_{min} < \omega_r < \omega_{max}$.

According to all the definitions stated above, the MPF problem can be stated as: given a trajectory $\mathbf{p}_t^I(t)$ with time derivative $\mathbf{v}_t^I$ and a desired geometric path $\mathbf{p}_d^T(\gamma)$ with desired speed $\dot{\gamma}$, the control problem is to design a control law for $\mathbf{u}(t)$ that steers the vehicle along the desired moving path $\mathbf{p}_d^I(t,\gamma) = \mathbf{p}_t^I + \mathbf{p}_d^T(\gamma)$ while satisfying the input constraints. Specifically, it is required to drive the term $||\mathbf{p}_r^I(t) - \mathbf{p}_d^I(t,\gamma)||$ towards an arbitrarily small neighborhood of the origin as $t \to \infty$.

3.2.1.2 Lyapunov-based Nonlinear Control Law

Following the definitions stated above, the error, $\mathbf{e}(t) = [\mathbf{e}_1\ \mathbf{e}_2]'$, can be defined as the difference between the vehicle current position and the desired virtual particle position to be followed, as $\mathbf{e}(t) = (R_R^I(t))'(\mathbf{p}_r^I(t) - \mathbf{p}_d^I(t,\gamma)) + \epsilon$, where ϵ is defined as a vector $\epsilon = [\epsilon_1\ \epsilon_2]'$ and it can be chosen to be arbitrarily small. By definition, the error is written with respect to the robot referential frame. The error dynamics are then defined in (3.2).

$$\dot{\mathbf{e}}(t) = (\dot{R_R^I}(t))'(\mathbf{p}_r^I(t) - \mathbf{p}_d^I(t,\gamma)) + (R_R^I(t))'(\dot{\mathbf{p}}_r^I(t) - \dot{\mathbf{p}}_d^I(t,\gamma)) \tag{3.2}$$

The terms $\mathbf{p}_d^I(t,\gamma) = \mathbf{p}_t^I(t) + \mathbf{p}_d^T(\gamma)$ and $\dot{\mathbf{p}}_d^I(t,\gamma) = \mathbf{v}_t^I(t) + \frac{\partial \mathbf{p}_d^T(\gamma)}{\partial \gamma}(\dot{\gamma})$ are the virtual particle position and its time derivative with respect to time, respectively.

Based on the definitions in (3.1) about the vehicle dynamics and the equations stated above, the error dynamics can now be rewritten as shown in (3.3).

$$\dot{\mathbf{e}}(t) = -S(\omega_r)\mathbf{e}(t) + \Delta\mathbf{u}(t) - (R_R^I(t))'\mathbf{v}_t^I(t) - (R_R^I(t))'\frac{\partial \mathbf{p}_d^T(\gamma)}{\partial \gamma}(\dot{\gamma}) \tag{3.3}$$

In the equation above, $\Delta = \begin{bmatrix} 1 & -\epsilon_2 \\ 0 & \epsilon_1 \end{bmatrix}$, where $\epsilon_1 \neq 0$ so that there is a direct control over the vehicle angular velocity.

The implemented control law based on Jain (2019) is represented in equation (3.4), where K_p is a constant known positive definite gain matrix and ϵ is defined such that the matrix Δ is invertible.

$$\mathbf{u}(t) = \Delta^{-1}(-K_p\mathbf{e}(t) + (R_R^I(t))'\mathbf{v}_t^I(t) + (R_R^I(t))'\frac{\partial \mathbf{p}_d^T(\gamma)}{\partial \gamma}\dot{\gamma}) \tag{3.4}$$

3.2.1.3 Proof of Convergence

Considering the Lyapunov function in equation (3.5), its derivative over time can be deduced as in (3.6).

$$V(\mathbf{e}(t)) = \frac{1}{2}\mathbf{e}(t)\mathbf{e}(t)' \tag{3.5}$$

$$\begin{aligned}
\dot{V} &= \mathbf{e}(t)'\dot{\mathbf{e}}(t) \\
&= \mathbf{e}'(t) - S(\omega_r)\mathbf{e}(t) + \Delta\mathbf{u}(t) - (R_R^I(t))'\mathbf{v}_t^I(t) - (R_R^I(t))'\frac{\partial \mathbf{p}_d^T(\gamma)}{\partial \gamma}(\dot{\gamma}) \\
&= -\mathbf{e}(t)'S(\omega_r)\mathbf{e}(t) - \mathbf{e}(t)'K_p\mathbf{e}(t) \\
&\leq -\lambda_{min}(K_p)||\mathbf{e}(t)||^2
\end{aligned} \tag{3.6}$$

In (3.6), $\lambda_{min}(K_p)$ denotes the minimum eigenvalue of K_p. Based on the derived result, the equilibrium particle $\mathbf{e}(t) = 0$ of the system is globally exponentially stable. It is also important to note that $||\mathbf{e}(t)||$ converges exponentially to 0 as time goes to infinity, implying that the moving path following error, $||\mathbf{p}_r^I(t) - \mathbf{p}_d^I(t,\gamma)|| \to \epsilon$ as $t \to \infty$ which means that the vehicle will converge to an arbitrarily small neighborhood of the virtual particle.

3.2.2 Virtual Particles Coordination

Now that a control law has been established to follow a given virtual particle in a desired moving path, the virtual particle needs to be defined so that coordination between vehicles is achieved. The coordination control law for controlling the rate of progression of the virtual particle along the parameterized path is based on the work done in Y. Wang et al. (2019b).

The control law implemented is presented in equation (3.7), where for each vehicle $i = 1, 2, ..., N$, $\tilde{\gamma}_i = \gamma_i - \gamma_{i-1} + \Delta_\gamma$ is the coordination error for vehicle i and Δ_γ is the desired along-path separation distance. $\beta \in \Re$ and $k_u \in \Re$ are positive constants. To achieve coordination, $\tilde{\gamma}_i \to 0$ must be assured as $t \to \infty$.

$$\dot{\gamma}_i = \dot{\gamma}_{i-1} - \beta\tanh(k_u\tilde{\gamma}_i) \tag{3.7}$$

It is assumed that there is a leader particle (particle number zero) that moves at a constant defined velocity (represented by $\dot{\gamma}_0$) inside the parameterized path. The first particle is then controlled via the equation in (3.7) and followed by UAV one, and so forth for every particle and UAV.

This solution implies a consecutive share of information between UAVs, namely the γ and $\dot{\gamma}$ parameters, with a distributed control strategy (see Figure 2.2). It is assumed that with this strategy, the necessary data is transmitted between UAVs, which is possible due to the fact that in this flight phase, the UAVs are relatively close to each other.

Proof of convergence

Let $W_i = \frac{1}{2}\tilde{\gamma}_i^2$ be a Lyapunov function candidate. Its derivative can be given by developing as done in equation (3.8).

$$\dot{W}_i = -\beta\tilde{\gamma}_i \tanh(k_u\tilde{\gamma}_i) \leq 0 \tag{3.8}$$

Since β and k_u are positive and the hyperbolic function $\tanh$ always has the signal of its input, the expression is always negative, implying and guaranteeing that $\tilde{\gamma}_i \to 0$ as $t \to \infty$.

3.3 Numerical Simulations

In the following section, numerical simulations are implemented based on the previously developed work in this chapter. The results that originated from these simulations are shown and discussed for validation of the model.

3.3.1 Implementation

In order to implement and validate the discussed model, architecture and controllers, the computing environment MATLAB (MathWorks (2022a)) was used together with its package based on graphical programming for modelling, Simulink (MathWorks (2022b)).

To start, the aircrafts needed to be modeled. For this case, it is assumed that the vehicles kinematics can be modelled as an unicycle type and that the linear velocity and the angular velocity can be directly controlled. This implementation resulted in the diagram shown in Figure 3.2. It has two inputs, the two control commands which are the vehicle's linear and angular velocity. This commands are then translated into the next position and attitude according to the considered unicycle kinematic model.

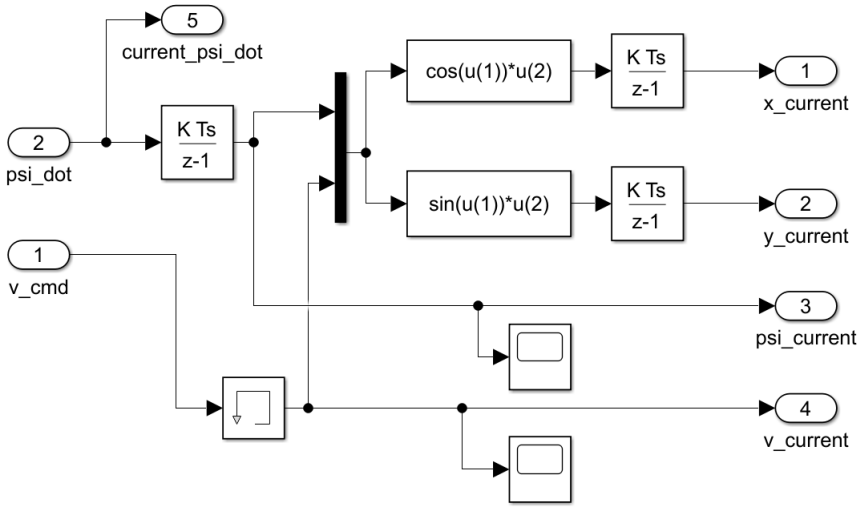

Figure 3.2: Simulink diagram for the vehicle model implemented.

The referred control inputs are computed via a script that implements the equations (3.4) and (3.7) addressed in Section 3.2 and sends them to the UAV. To compute this control commands, several inputs are necessary.

The first input for the implemented control law is the aircraft state, more specifically, the aircraft position, velocity and attitude. It is also necessary to input the target position and velocity which, at this stage, is simulated as a simple block that simulates the kinematics of the target and sends them to the controllers. The last input necessary is the particle position and velocity along the desired circular path, the γ and $\dot{\gamma}$ shown in Subsection 3.2.2. The scheme of the implementation in Simulink of this control architecture is depicted in Figures 3.3 and 3.4.

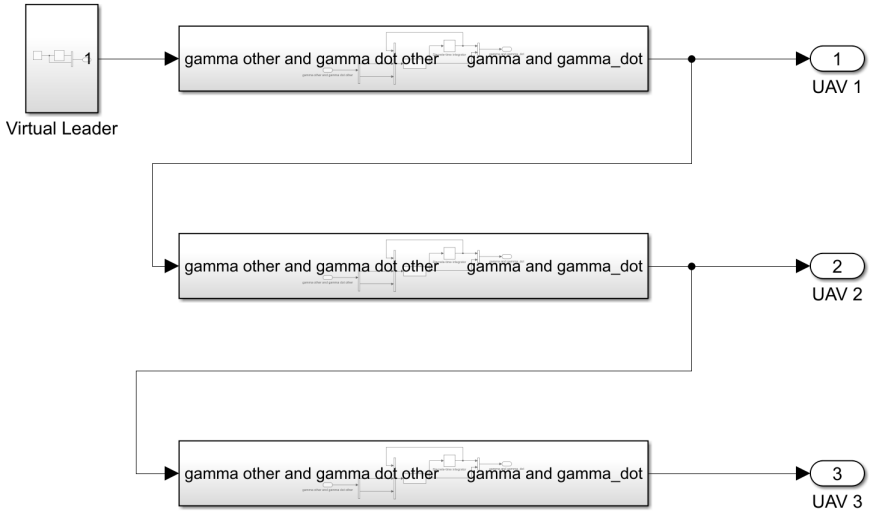

Figure 3.3: Simulink diagram scheme of the control architecture for the virtual particles.

The control starts with the generation of a virtual leader particle, which outputs a constant defined $\dot{\gamma}$ and the γ computed in each instant of time. For each of the blocks depicted in Figure 3.3, there are two inputs and two outputs: the γ and $\dot{\gamma}$ of the particle that it is following and the γ and $\dot{\gamma}$ of the particle itself. To compute the $\dot{\gamma}$ in the next instant of time, the equation (3.7) is implemented in a script, represented by the Interpreted MATLAB Fcn block in Figure 3.4.

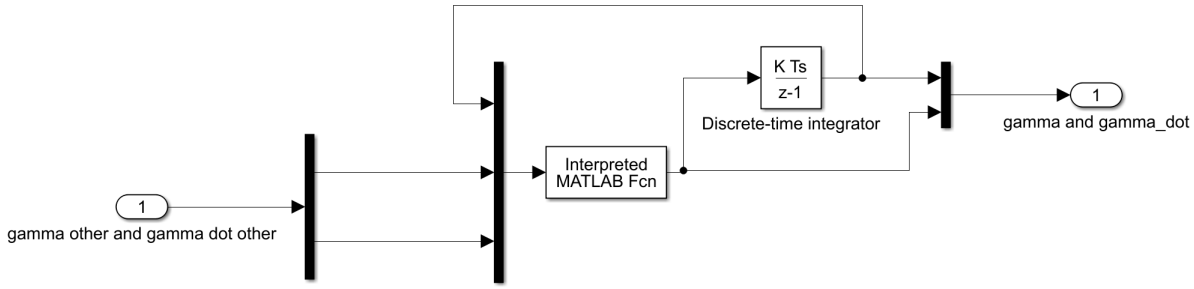

Figure 3.4: Simulink diagram scheme of the inside of one of the blocks depicted in Figure 3.3.

The final implementation can be seen in the diagram shown in Figure 3.5, where the red block represents the target kinematics, the green block represents the control architecture that defines the particles positions and velocities, the grey blocks represent each of the simulated aircrafts and lastly, the blue block, that allows the drawing of the live results and show the final desired plots.

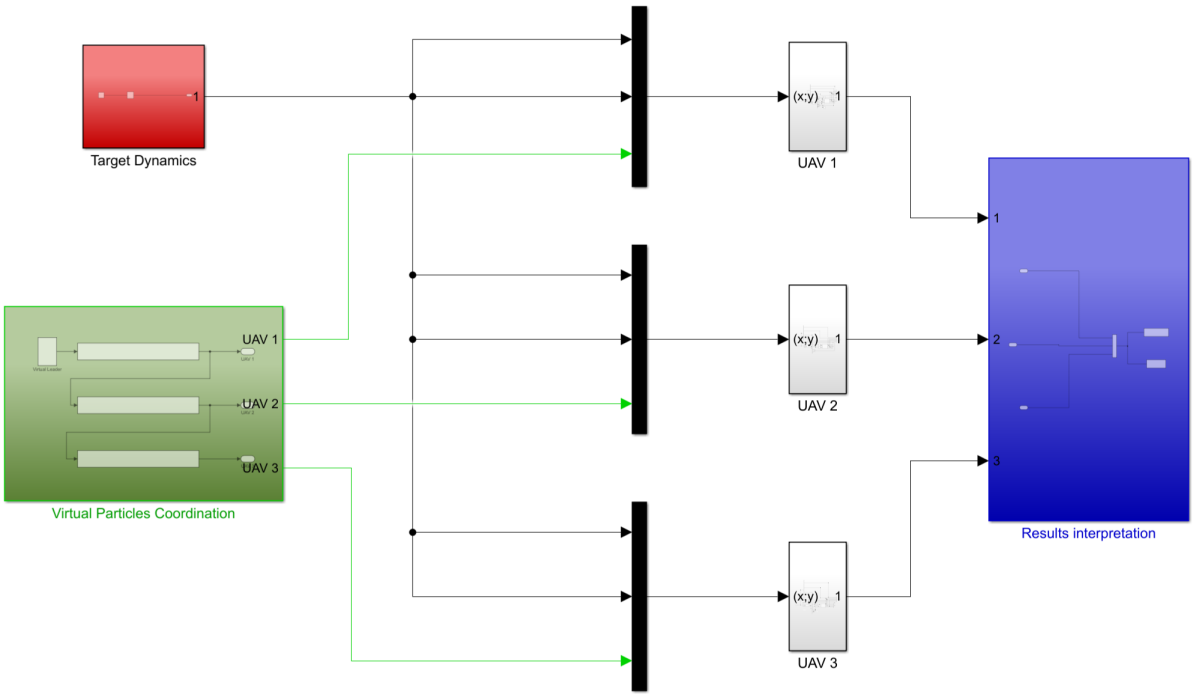

Figure 3.5: High-level view of the implemented Simulink architecture.

3.3.2 Verification and Validation

In this section, the results are presented after the implementation of all the necessary parts to simulate the model and the proposed control architecture. The results shown refer to two simulations ran with the properties found in Table 3.1. In every plot shown, there is a colour correspondence between the graphs and the UAVs: red, green and blue indicate the results regarding UAV one, two and three, respectively. The starting positions for each aircraft are $\mathbf{p}^I_{r_1}(0) = [-200;\ -200]$, $\mathbf{p}^I_{r_2}(0) = [-300;\ -200]$, $\mathbf{p}^I_{r_3}(0) = [-400;\ -200]$ and the target starts at the position $\mathbf{p}^I_t(0) = [0;\ 0]$, in meters, all according with the inertial frame and all with 0 radians starting heading.

Simulations Properties	
Sampling Frequency	20 Hz
Step Size	0.05 s
K_p	[0.9 0; 0 0.05]
β	2
k_u	0.1
ϵ	[1; 0]
$\dot{\gamma}_0$	15 m/s
$\gamma_i(0)$	$-i\Delta_\gamma$ m
r	200 m
Δ_γ	$\frac{2\pi r}{N}$ m
N	3

Table 3.1: Simulation properties for the results shown.

3.3.3 Steady Target Simulation

In this first simulation, the numerical simulation ran for a total of two hundred seconds. By analysing the Figure 3.6, it is possible to conclude that the aircrafts assumed a circular movement around the target with a desired radius of two hundred meters. It can also be deduced visually that the particles finished with the desired distances between each other, as also proved by the graph in Figure 3.11. The target remained steady for the whole simulation.

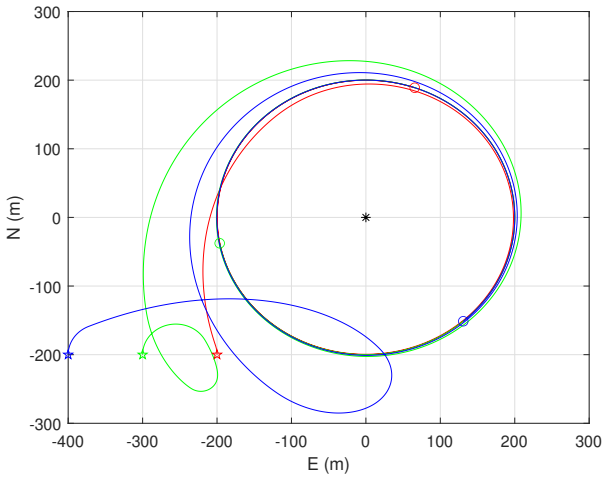

Figure 3.6: Path drawn by each UAV. The black star represents the target position, the coloured stars represent the starting points and the coloured circles represent the ending points.

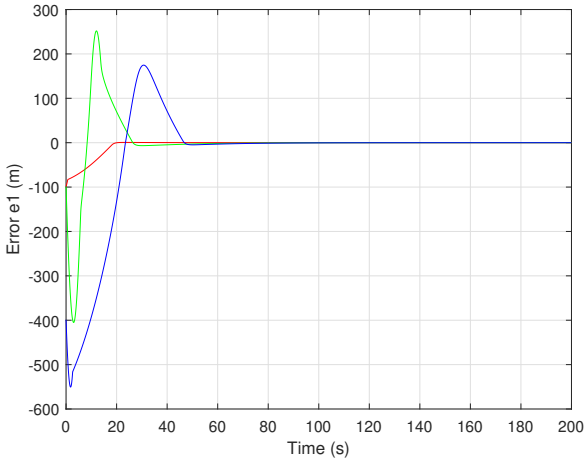

Figure 3.7: Error $\mathbf{e}_1$, as described in Section 3.2.1.

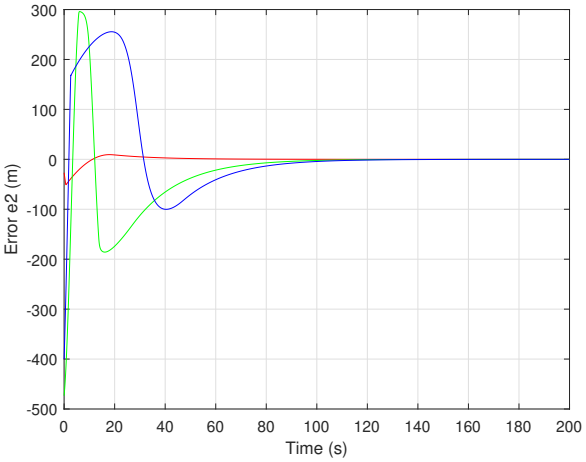

Figure 3.8: Error $\mathbf{e}_2$, as described in Section 3.2.1.

The graphs depicting the evolution of the tracking error over time (Figures 3.7 and 3.8) show that all the aircrafts converge to the respective virtual particle on the desired path.

The commanded control inputs converge to the wanted values: 15 meters per second for the commanded velocity and a heading rate given by the desired velocity over the radius, $\omega_r = \frac{15}{200} = 0.075$ radians per second (or approximately 4.3 degrees per second). It is also important to notice that the saturated behaviour when the commands are too high or too low are due to imposed limitations for a more realistic simulation: the aircrafts max-

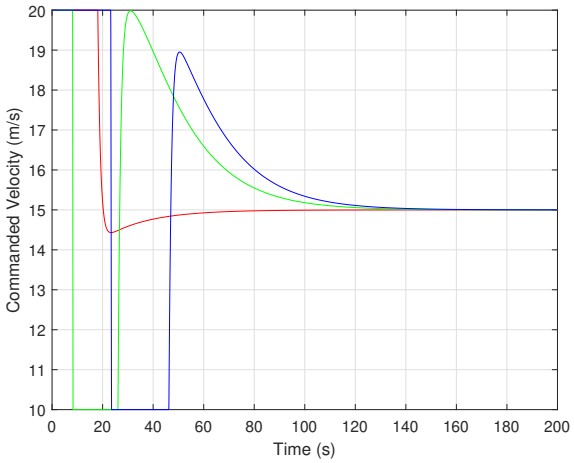

Figure 3.9: Commanded velocity.

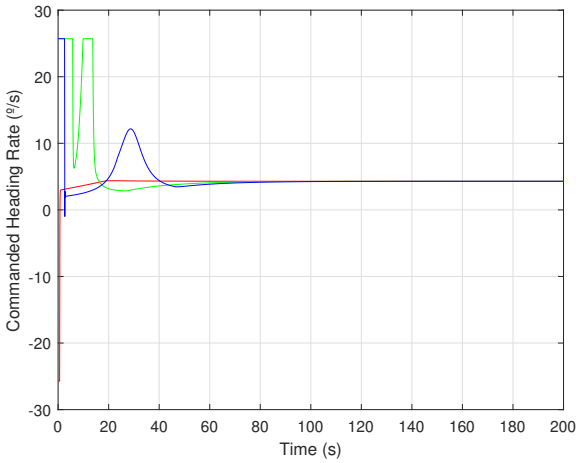

Figure 3.10: Commanded heading rate.

imum and minimum commanded velocity are set to 20 and 10 meters per second, respectively and the maximum and minimum commanded heading rate possible are set to $\frac{\pi}{7}$ and $-\frac{\pi}{7}$ radians per second, also respectively.

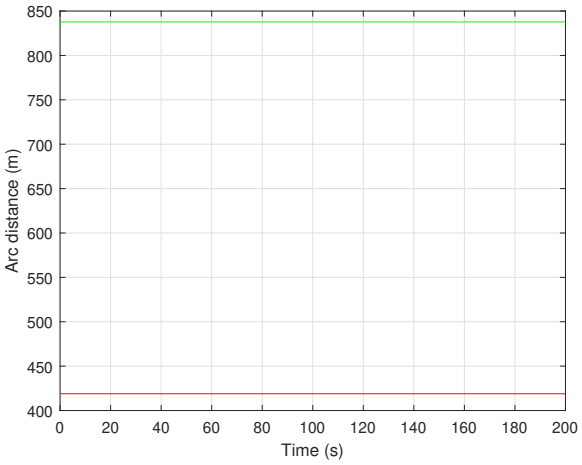

Figure 3.11: Distance between particles.

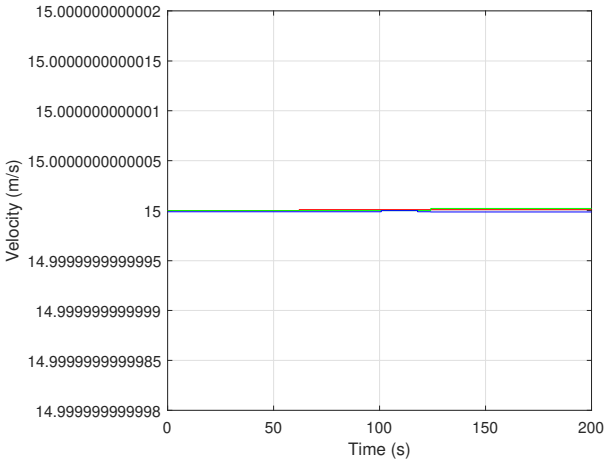

Figure 3.12: Particles velocities in the circular path.

Regarding the implemented architecture to set the position of each virtual reference particle, the graph in Figure 3.12 shows that the particles kept a desired speed of $\dot{\gamma} = 15$ meters per second during all the simulation. The graph shown in Figure 3.11 expresses the distances between particles one and two in red and between particles one and three in green; this distances remain at the desired values since the desired distance between consequent particles is given by $\Delta_\gamma = \frac{2\pi r}{N}$, which in this particular simulation is approximately 418.87 meters measured in the arc of the circumference. Since the third particle is lagging $\Delta_\gamma = 418.87$ meters behind particle number two, the distance between itself and particle number one is two times the desired distance, approximately 837.76 meters.

From the obtained results, one can conclude that the implemented model and controller are correctly implemented for a steady target since the particles were successfully controlled to move at a desired speed and keep a defined distance as the aircrafts were steered to follow them. In summary, the obtained results show that the

aircrafts followed the target around a circular path while maintaining equal distances between each other.

3.3.4 Moving Target Simulation

Since the ultimate goal is to have the UAVs follow a moving target in a coordinated way, this subsection shows the results for simulations where the target is moving with a specified acceleration and heading rate; these are given by $||\dot{\mathbf{v}}_t^I||(t) = 0.1\sin(0.07t)$ meters per second squared and $\omega_t(t) = 0.02\cos(0.03t)$ radians per second (based on Oliveira, Aguiar, and Encarnacao (2016)). The target motion had to be carefully chosen since the vehicle velocity can not go beyond a certain limit imposed by the maximum aircraft speed, which in this case was set to 20 meters per second; the target velocity limit however is lower than 20 meters per second, since the aircraft still needs to able to follow the particle in every time instant. Being so, the target maximum velocity limit is bounded by the difference between the aircraft maximum velocity and the particle velocity in the path, $||\mathbf{v}_t^I||(t) < 20 - 15 = 5$ meters per second.

The simulations ran for a total of 200 seconds, enough simulation total time to have the target describe a considerable and easily visible movement.

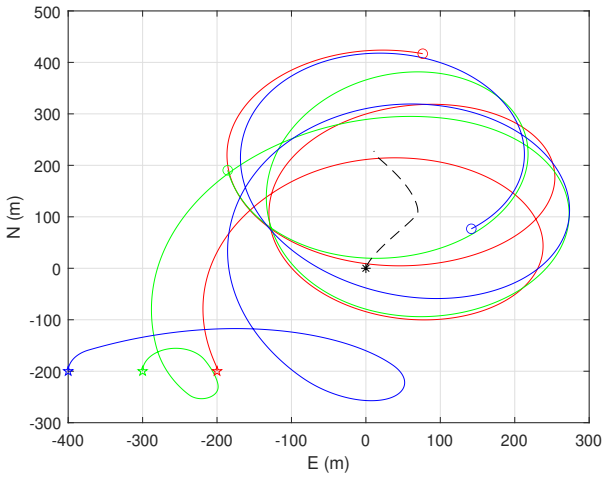

Figure 3.13: Path drawn by each UAV and by the target (in a dashed black line). The black star represents the target initial position, the coloured stars represent the starting points and the coloured circles represent the ending points regarding the UAVs movement.

Regarding Figure 3.13, it is possible to characterize the target path as curve where the target itself changes both in orientation and in velocity through time. The starting points for the aircrafts and the target were set to be the same as in the steady target simulations. The trajectory described by the aircrafts is the desired one, always forming a circle around the moving target, with an equal distance inside the circular path.

The errors plotted in Figures 3.14 and 3.15 converge to zero as time goes to infinity, as expected, similar to the results obtained with the steady target simulation.

The commanded values for each aircraft show expected results since in this case, the aircrafts need to constantly change their heading rates and velocities to achieve coordination and stay inside the path around the moving target. With the exception of the initially sent commands (until 60 seconds of simulation time), after all UAVs converge to the corresponding virtual particle, the behaviour is smooth and the values oscillate around the expected desired

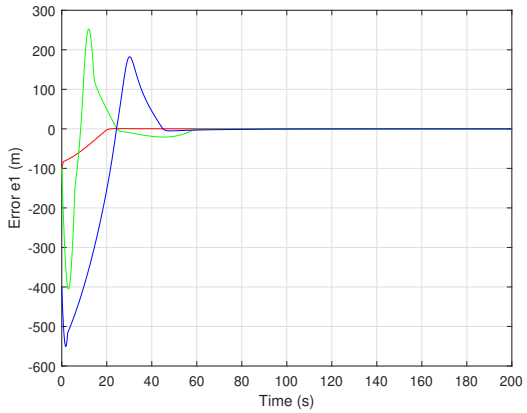

Figure 3.14: Error $\mathbf{e}_1$, as described in Section 3.2.1.

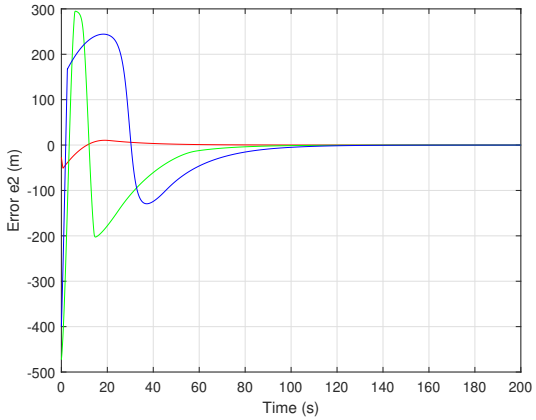

Figure 3.15: Error $\mathbf{e}_2$, as described in Section 3.2.1.

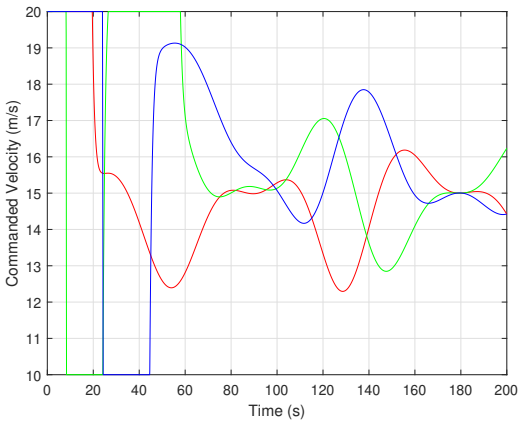

Figure 3.16: Commanded velocity.

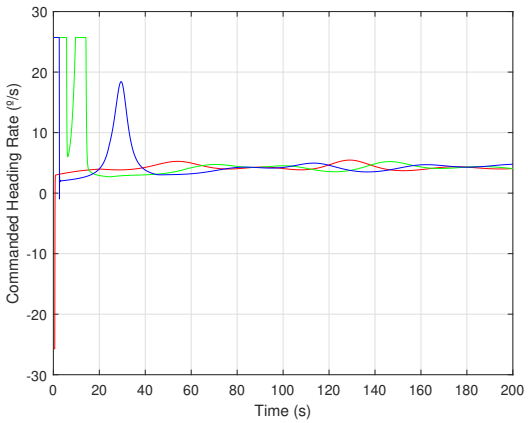

Figure 3.17: Commanded heading rate.

velocity and heading rate if the target was stationary: 15 meters per second and approximately 4.3 degrees per second, as computed in Subsection 3.3.3.

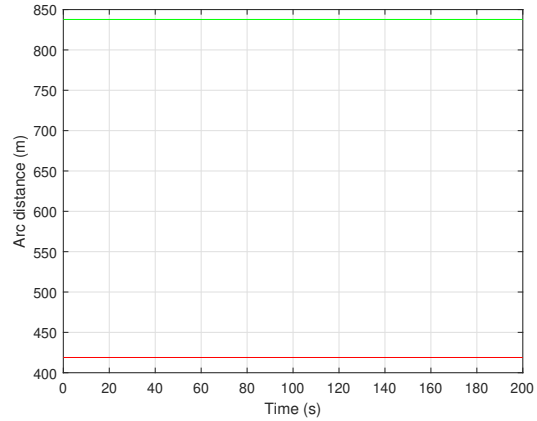

Figure 3.18: Distance between particles.

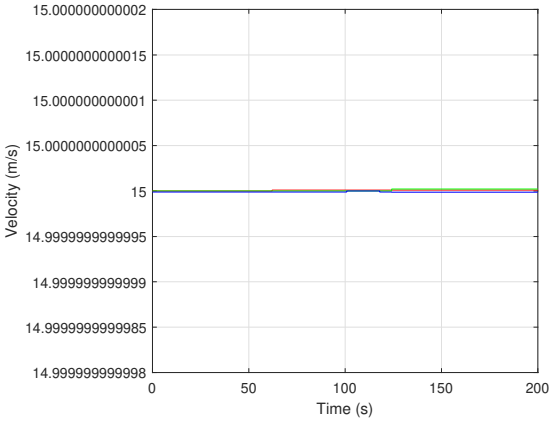

Figure 3.19: Particles velocities in the circular path.

Regarding the results obtained from the particles control implementation, shown in Figures 3.18 and 3.19, there is no difference when compared with the graphs obtained in the first simulation. This can be simply justified by the fact that this control is independent of the target movement and it depends only on the distance wanted between the aircrafts, the circular path radius, the desired speed and the control gains, β and K_u.

Since the model and its implementation could be validated through the results shown previously, the implementation continues by setting up a more realistic environment, as addressed in the whole fourth chapter.

Chapter 4

Software In The Loop Simulations

After concluding the numerical simulations presented in the previous chapter, it is necessary to move to the implementation and simulation in a more realistic environment. To achieve this, an open source software stack was deployed and used to both develop the control architecture and test it in a software in the loop simulation. This chapter explains in detail how everything was connected together to run this important step in controller design and architecture validation.

This chapter starts by providing a general overview of what software is used and what it is used for. Then, it moves to the actual implementation of the proposed architecture and finishes with the results of such implementation.

4.1 Software Architecture

This section aims to describe each software application module necessary to run the software in the loop simulations. The selected software and architecture are heavily based on the work developed in Alves et al. (2022).

4.1.1 Operative System

In order to make the best use of the already available resources, a machine with the operative system Ubuntu 20.04 was used since there is several documentation available and most of the open source software needed to run this type of simulations is made to be ran in this operative system. The machine specifications are the ones found in Table 4.1.

Machine Specifications	
Operative System	Ubuntu 20.04.5 LTS 64-bit
Memory	31.3 GB
Processor	i7-4820K CPU @ 3.70Ghz 8
Graphics	NVIDIA Corporation GP106 [GeForce GTX 1060 6GB]

Table 4.1: Used machine specifications.

4.1.2 Autopilot Controller

In CIAFA's adopted control system architecture, the aircraft attitude and movement cannot be directly commanded via sending the control inputs derived in the previous chapter. The velocity and heading rate commands stated previously have to be converted to actual usable command inputs for the aircrafts inner loop control systems as the thrust or the deflection of control surfaces such as the ailerons, the rudder or the elevator (see Figure 4.1). These conversions are done using an controller available from open sources. In this case, CIAFA's control system architecture relies on PX4 (Meier, Tanskanen, Fraundorfer, and Pollefeys (2011)).

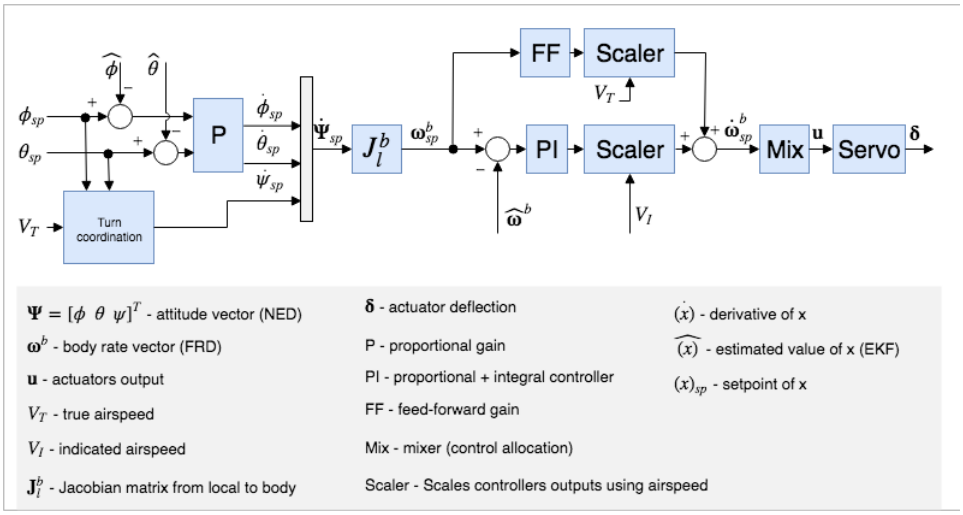

Figure 4.1: Diagram of the attitude controller used by PX4 (retrieved from PX4 (2022)).

In the representative scheme in Figure 4.1, the final output are the control surfaces deflections necessary to achieve a given attitude, velocity or positions, which are usually defined by the user as the input to PX4.

4.1.3 Simulation Environment

To simulate the physical environment with the necessary models, such as the aircrafts or the runway, the open source three dimensional simulator Gazebo was used (Koenig and Howard (2004)).

Gazebo allows the user to create models and worlds or use previously made available ones, such as the ones provided by the PX4 repository (PX4 (2022)). In this repository, the plane model (which is modeled based on SIG Rascal UAV) was the chosen aircraft since it simulates a fixed-wing model, can carry an onboard camera and is commonly used in this type of simulations, being similar in mass and volume to the UAVs used in CIAFA (see Figure 4.2). There are also several available worlds, which can be used as a basis to build a more customized one. In this case, the "empty" world file was modified to include the spawn of the a grassy terrain and a runway, as well as a model car to serve as the target to be followed (see Figure 4.3).

4.1.4 Ground Control Station

In order to create a virtual ground control station and send the initial commands to the aircrafts such as arming or mode changing, QGroundControl (QGC) was the program selected and used (LambDrive (2022)). This powerful and intuitive ground station simulator allows the connection of multiple vehicles and supports any autopilot

Figure 4.2: Side by side comparison between SIG Rascal (on the left) and ANTEX X02 - Alpha (on the right), one of the UAVs used by CIAFA.

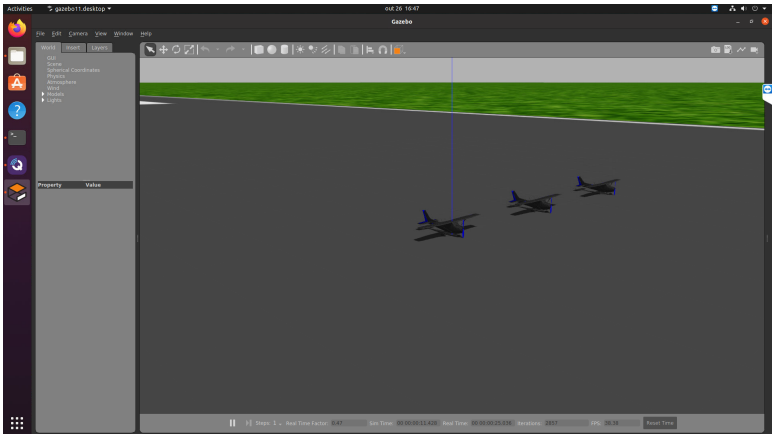

Figure 4.3: Screenshot of Gazebo's user interface with the customized world file simulated.

controller that uses MAVLink protocol (QGC (2022)), for instance, PX4.

4.1.5 Communication Protocol

MAVROS is a ROS package that enables the communication between all the software mentioned previously. It works for any MAVLink enabled autopilot, ground station, or peripheral, which is the case for the chosen ones: PX4, QGC and Gazebo.

As it is shown in the Figure 4.4, all the communication systems use the MAVLink protocol. All the telemetry from the UAVs is transmitted from the autopilot controller, PX4, to the environment simulator, in this case Gazebo, so that there is a real world simulation happening in real time with the simulated movement of the aircraft. The QGC block establishes a simulated ground control station sending and receiving data to and from the autopilot controller. The ROS Environment block is where all the written and implemented ROS nodes are integrated through ROS and it is further explained in detail in the next section.

4.1.6 ROS Environment

The Robot Operating System (ROS) is a set of software libraries and tools that help build robot applications. The selected version to work with was ROS 1 most recent distribution, ROS Noetic Ninjemys (ROS (2022)).

It is important to define concepts that will be used throughout the implementation. A node is a process that performs some assigned computational task. They are usually combined together with other nodes using topics and

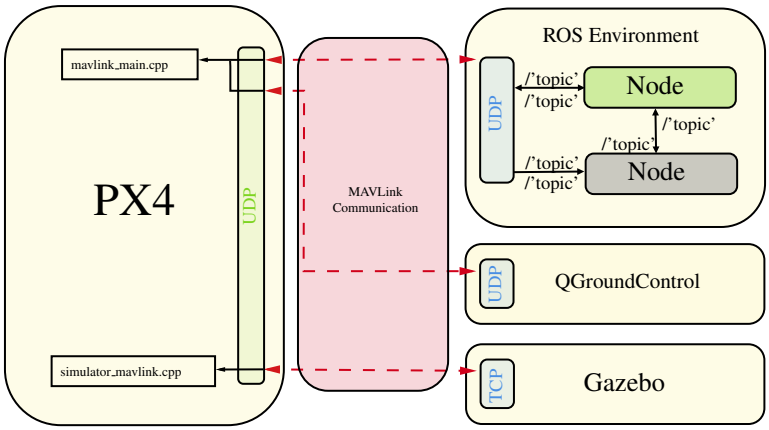

Figure 4.4: General standard scheme depicting the essential simulation in the loop software and its connections.

services (ROS (2022)). A topic is a name bus used by nodes to exchange messages and it can be either subscribed or published depending if the node needs to fetch or send information from the topic, respectively (ROS (2022)). The data contents are sent to the topics through defined messages, which is a simple data structure, comprising typed fields (ROS (2022)).

To implement the simulations, several nodes, topics and messages had to be defined and coded into the ROS environment while others were already implemented and defined by the ROS environment itself and by PX4. A more detailed explanation on this implementation can be found in the sequel.

4.2 Control Architecture

After the introduction of all the necessary software tools to simulate a coordinated flight, this section aims to show how everything is connected. It starts by explaining the single vehicle architecture and then moves on to the architecture regarding having several vehicles.

4.2.1 Single UAV Control Architecture

For a single vehicle, the architecture can be structured as presented in Figure 4.5. There are three modules inside the main block: the offboard controller that steers the vehicle to the desired virtual particle, the controller that decides and computes the particle position and velocity along the path and the target data publishing module. This are implemented as nodes in the ROS environment: one node for the leader particle, one for the particle controller, one for the vehicle controller and one whose function is to constantly publish the target position and velocity.

All the published and subscribed topics are represented in every drawn connection. In Table 4.2, the nodes used to publish and subscribe this topics, as well as their number of instantiations are described.

4.2.1.1 Vehicle Controller

The main goal for this controller is to steer the UAV in the desired direction, based on the information given about the aircraft, the target and the corresponding desired virtual particle. The inputs for this node (red dashed

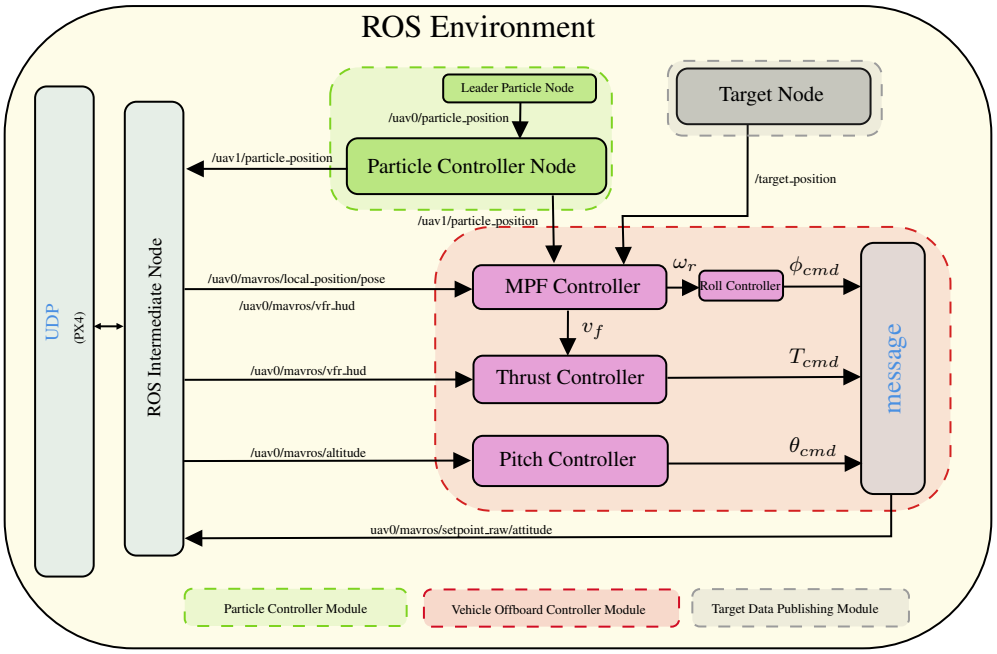

Figure 4.5: Scheme depicting the single vehicle control architecture.

Node	Number of instantiations
Target	1
Leader Particle	1
Particle Controller	1 for each UAV
Vehicle Controller	1 for each UAV

Table 4.2: Nodes used and number of instantiations.

block in Figure 4.5) are the target position and velocity, the UAV current attitude, position and velocity and the particle to be followed velocity and position along the path.

The information about the target is computed and sent through a publishing node implemented in the target node block. This node publishes a message with all the data on the target such as its position, attitude, velocities and accelerations. The only information used by the controller is the current target position and velocity, as described previously in Chapter 3.

The UAV data is received from the nodes provided by the PX4 autopilot controller in the ROS environment, which were available in ROS Noetic package documentation in ROS (2022). The processes inside the vehicle controller block in Figure 4.5 take this information and compute the necessary MPF commands to converge to the desired references.

The output of the MPF controller is the commanded attitude to be sent to the autopilot controller, which is sent as a single message with the commanded pitch, roll and thrust computed to achieve the desired velocity and heading rate. Since the control cannot be done by directly sending the linear and angular velocity commands (as opposed to what was done in the numerical simulations in Chapter 3), this control commands need to be converted to actually usable information to be sent to the autopilot. From the topics available and already implemented by PX4, only one is able to control the fixed wing plane model attitude, having two different possible message types differing in the input combinations allowed (PX4 (2022)). The chosen input combination consists in a attitude quaternion and

a thrust value. Due to this setup, an extra implementation (compared with the numerical simulation) was necessary so that the computed commands in the originally defined vehicle controller could be used by the autopilot's inner loop controller. This conversions are done using a software module developed in Alves et al. (2022), which is briefly described next.

Starting by the commanded velocity, this value needs to be converted into a thrust command to be sent to the aircraft. To achieve this transition, a Proportional-Integral (PI) controller was designed (as shown in Figure 4.6) and implemented through the function represented by the thrust controller block in Figure 4.5, where the input is the desired velocity computed by the implemented vehicle controller and the output is the necessary thrust command to achieve said velocity. The values $k_p = 0.3$ and $k_i = 0.2$ were set to be the proportional and the integral gains, respectively.

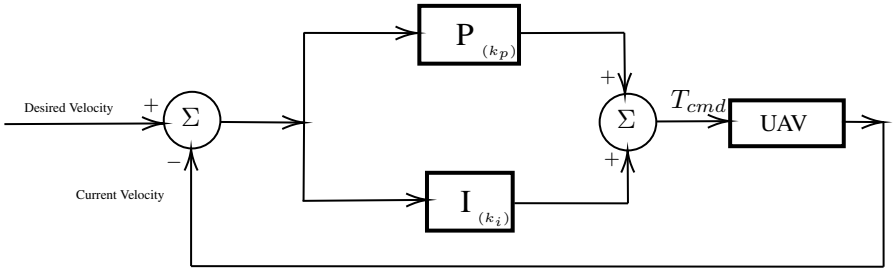

Figure 4.6: Thrust PI controller representative scheme.

The heading rate command is then converted into a roll command. This command is then limited to a certain absolute value to avoid flight instability, keeping the commands between reasonable and feasible angular rates. The formula that allows the roll command computation based on the heading rate command is depicted in equation (4.1) (McLean (1990)), where r is the path radius, $g = 9.81$ is the Earth's gravity acceleration and U_0 is the UAV's airspeed.

$$\phi = \arctan(\frac{U_0^2}{rg}) = \arctan(\frac{\dot{\psi} U_0}{g}) \tag{4.1}$$

Since the environment is now in three dimensions, as opposed to the simulations ran in the numerical simulations which were two dimensional, it is necessary to add a longitudinal control component to control the altitude at which it is desired for the aircrafts to fly in. To this end, a simple PI controller was implemented to control the UAV's pitch based on the error in altitude, as graphically presented in 4.7. The gains were set to be $k_p = k_i = 0.2$ for both the proportional and the integral gain. The controller and its computations are represented by the pitch controller block in Figure 4.5.

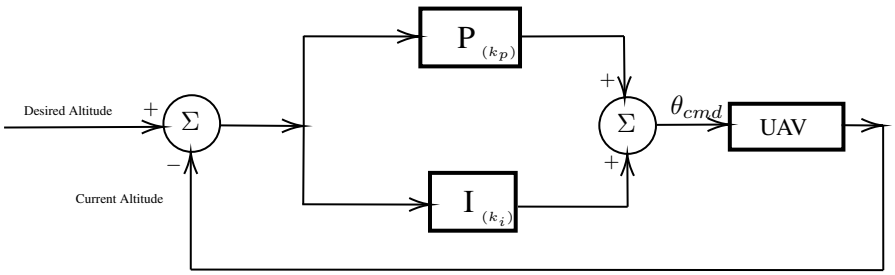

Figure 4.7: Pitch PI controller representative scheme.

After all the necessary values for the aircraft's orientation have been explicitly computed, this are formatted as an attitude quaternion, so the message can be sent to the attitude controller topic with the required format, together with the desired thrust value. The process described is represented by the message block inside the vehicle controller node in Figure 4.5. The message publishing is implemented via the vehicle controller node, after calling all the processes presented above using the periodically subscribed data.

4.2.1.2 Particle Controller

Regarding the particle controller (represented in the green dashed box in Figure 4.5), the input is the particle $i-1$ data (position and velocity along the path desired moving path) and it outputs the data on the i particle (similar to what was implemented in the numerical simulations). For the leader particle, an independent node was created that takes no inputs and only outputs the data on particle number 0, represented by the leader particle node block in Figure 4.5. The remaining particles are controlled via the nodes implemented in the particle controller node blocks.

With the received information and using the equations presented in the previous chapter, the velocity of the particle in the path is commanded for each iteration by $\dot{\gamma}$, governing its movement. Finally, the output is the vehicle own particle data, which is then transmitted to the next aircraft in the chain.

4.2.2 Multi-UAV Coordination Architecture

In order to achieve a coordinated flight with several UAVs, it is necessary to define the architecture behind the connections between the vehicles and how the controllers are integrated. The ROS Environment block in Figure 4.4 is now composed by several repeating processing nodes, as represented in Figure 4.8. The addition of each new aircraft adds a new single vehicle block (Figure 4.5), and it can be easily scaled up to ten UAVs (see Section 4.4 on practical considerations). The only two nodes that do not replicate with each new UAV are the target kinematics publishing node (represented by the grey target node block), which publishes to a topic that is used by every vehicle, and the leader particle node, which is subscribed only by the first UAV.

The node created to publish the data on the leader particle is only instantiated once, since there is only one leader particle, as it is represented in Figure 4.8. For the remaining UAVs, the particle controller and the vehicle controller nodes are repeatedly instantiated for each new aircraft added, forming a structural scalable block represented by the cyan colour with the UAVN tag.

Once again, all the defined topics to send and receive messages through nodes are represented in the representative scheme in Figure 4.8. It is important to notice that for the newly added UAV1, all the topics are different (with the exception of the target position topic) and the update is based on the UAV number.

4.3 Simulation Results

In this section, the results for the software in the loop simulations with the proposed architecture implemented are presented.

To start the simulations, a *launch* file that initiates all the necessary nodes was created. The instantiated nodes are the leader particle node, the target publishing node and the UAV controller nodes, one for each new UAV added.

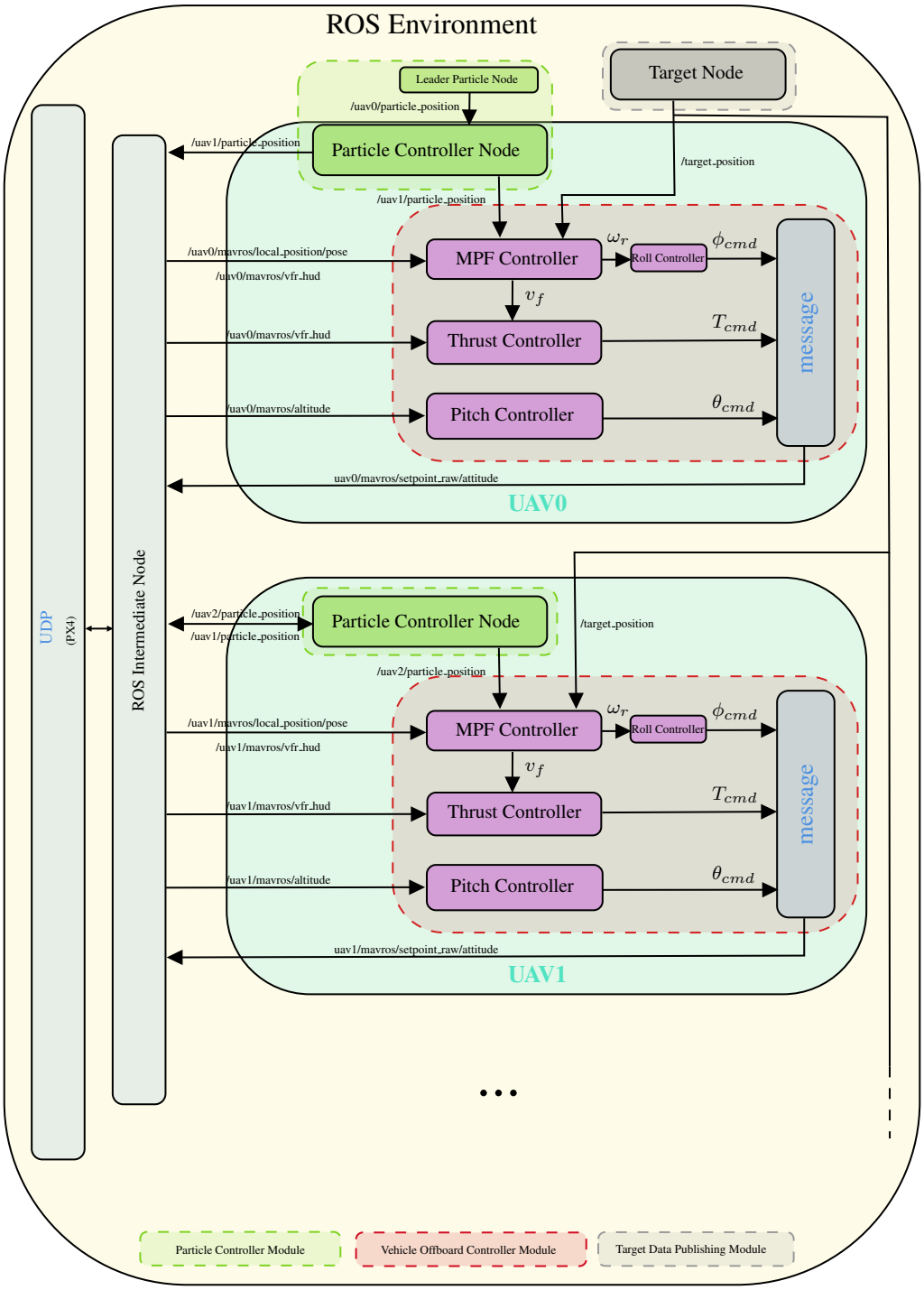

Figure 4.8: Multi-UAV architecture and connections scheme.

To use this file, a prebuilt tool for easily launching multiple ROS nodes locally (ROS (2022)) is used. This tool connects with the PX4 database on models and worlds and instantiates the specified nodes.

For the results presented in the next section, three UAVs were used and implemented, similarly to what was done in the numerical simulations implementation. Differently to what was done in the numerical simulation in Chapter 3, the simulations start with the UAVs on the ground and the starting positions for each one of them was chosen to be $\mathbf{p}^I_{r_1}(0) = [0;\ 2]$, $\mathbf{p}^I_{r_2}(0) = [0;\ 4]$ and $\mathbf{p}^I_{r_3}(0) = [0;\ 6]$ with the target starting at the same position $\mathbf{p}^I_t(0) = [0;\ 0]$. In terms of simulations time, the plotted figures start at the moment that the aircraft first enters

offboard mode and takes off and finish at the shutting down time of the simulation. First, the results for a steady target are presented, followed by the results regarding following a moving target.

The values used for the software in the loop simulations can be found in Table 4.3. There are some differences when compared to the values used in the numerical simulations, namely the MPF controller gains, β and the ϵ values. This had to be adjusted due to the UAVs' dynamics properties added by the inner loop controllers. Reducing the gain matrix proved to give better, smoother results. The value of ϵ, which can be seen as an error tolerance, was increased, allowing for the system to "relax" over the error value oscillations, which avoided saturation in the sent control commands.

Simulations Properties	
Control Rate Frequency	20 Hz
K_p	[0.09 0; 0 0.005]
β	0.2
k_u	0.1
ϵ	[10; 0]
$\dot{\gamma}_0$	15 m/s
$\gamma_i(0)$	$-i\Delta_\gamma$ m
r	200 m
Δ_γ	$\frac{2\pi r}{N}$ m
N	3
Altitude	30 m

Table 4.3: Simulation properties for the results shown.

In all the presented results, the graphs are displayed with the vehicles correspondent colours: red refers to the first UAV, green to the second UAV and blue to the third UAV. The same goes for the particle data presented, as it was done in Chapter 3. The stars represent the UAVs starting positions and the circles the final positions, either for the UAVs or for the target.

4.3.1 Steady Target

In Figure 4.9 it is possible to see the trajectories of each UAV during the simulation. From a two dimensional perspective (the North-East plane), the path drawn is similar to the one obtained in the numerical simulations, as expected and shown in Figure 4.10. When adding a third dimension, it is possible to see that after taking off, all the UAVs ascend to an altitude of 30 meters, as desired, reaching it approximately after 20 seconds, and maintaining this behaviour during the whole simulation, as depicted in Figure 4.11.

Figures 4.12 and 4.13 show the UAVs and target trajectories up until 75 and 150 seconds, respectively. The black dashed circle represents the circular path around the target, in which the vehicles should fly in, equally distanced from each other. The tendency is clear, the UAVs converge and maintain themselves in the circular path, a non moving reference around the steady target. The next presented figures explore in detail the most important variables to assess the control system performance, such as the errors or the commanded control inputs, and how they change through time.

The errors plotted in Figures 4.14 and 4.15 converge to a bounded value close to 0 as time goes to infinity, with the presence of a static error, which was not present in the numerical simulations, as shown throughout the

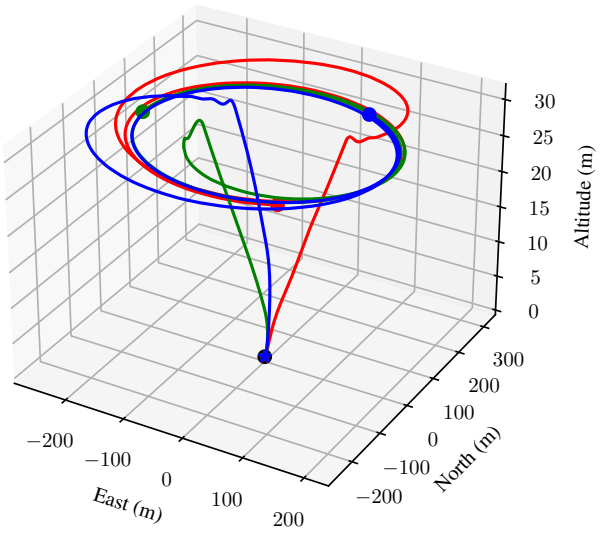

Figure 4.9: UAVs and target paths.

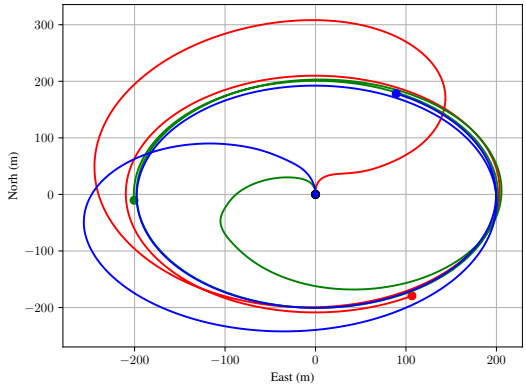

Figure 4.10: Longitudinal (North-East plane) perspective of the UAVs and target paths.

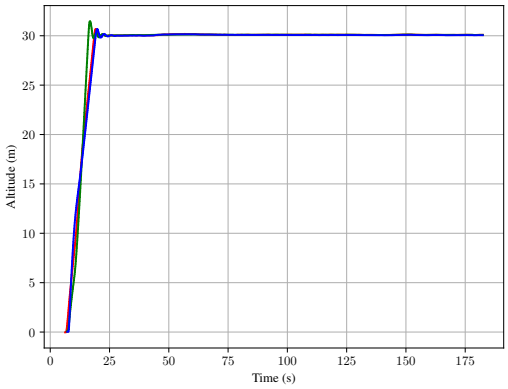

Figure 4.11: UAVs altitude (in meters) over time (in seconds).

results in Chapter 3. In the software in the loop simulations, the model dynamics are changed, mainly due to the introduction of inner loop controllers, which subsequently leads to delayed commands, causing the presence of a static error. Considered solutions can be found in Section 4.4. Regarding the error $\mathbf{e}_1$, although it has some small amplitude oscillations, it converges to a value around 0, as desired. Error $\mathbf{e}_2$ has lower amplitude on the oscillations and the evolution through time is similar to the one found in $\mathbf{e}_1$.

By analysing the graphs in Figures 4.16 and in 4.17 it is possible to conclude that there is a tendency for the values to stabilize around the desired command input. In the case of the commanded velocity, it is expected that, in order to follow the circular path around the steady target at the defined $\dot{\gamma}_0 = 15$ meters per second, the computed and sent value stabilizes in the 15 meters per second. In the plots from Figure 4.16 it is clear that the value indeed

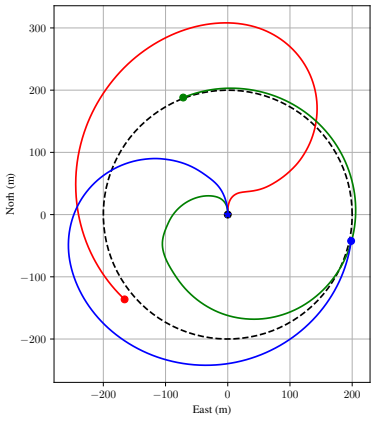

Figure 4.12: Longitudinal perspective at 75 seconds.

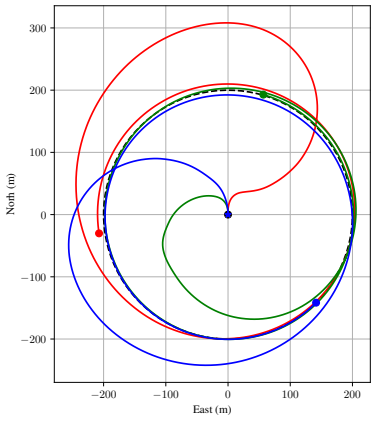

Figure 4.13: Longitudinal perspective at 150 seconds.

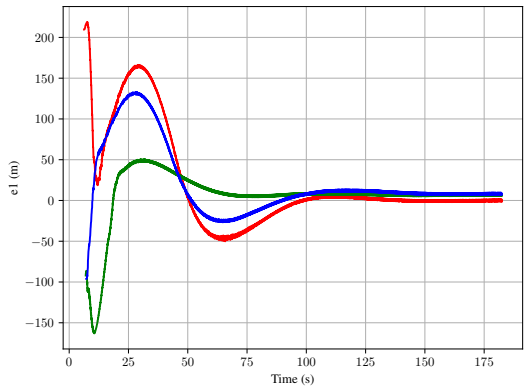

Figure 4.14: Error $\mathbf{e}_1$, as described in Section 3.2.1.

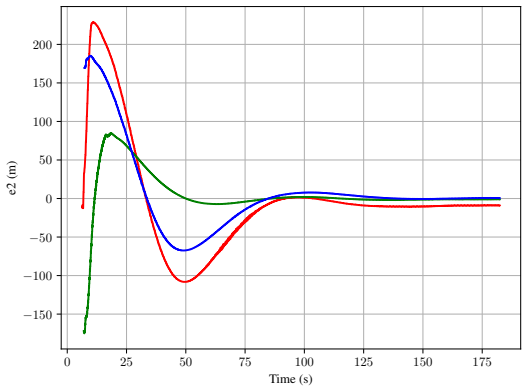

Figure 4.15: Error $\mathbf{e}_2$, as described in Section 3.2.1.

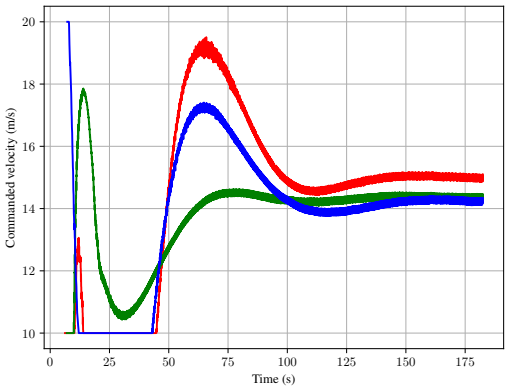

Figure 4.16: Commanded velocity.

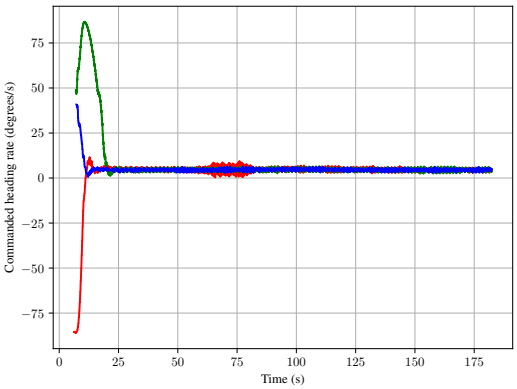

Figure 4.17: Commanded heading rate.

tends to around 15 meters per second, as desired, although with some low amplitude oscillations. The same goes for the commanded heading rate, it is expected that, to achieve the desired coordinated flight, the value reaches

and stabilizes at 4.3 degrees per second, as deduced in Chapter 3. In the implemented solution, although there are some oscillations in the sent command, they occur around the desired value, allowing the vehicle to steer to the desired direction and leading the error to values close to 0, as previously shown.

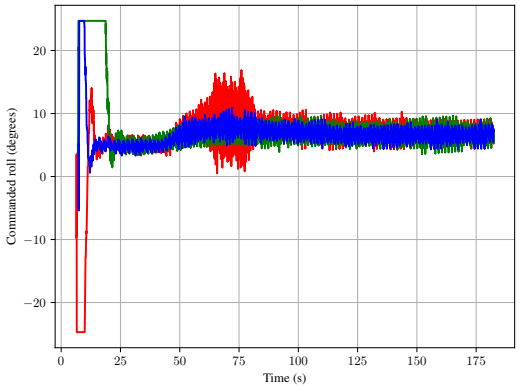

Figure 4.18: Commanded roll.

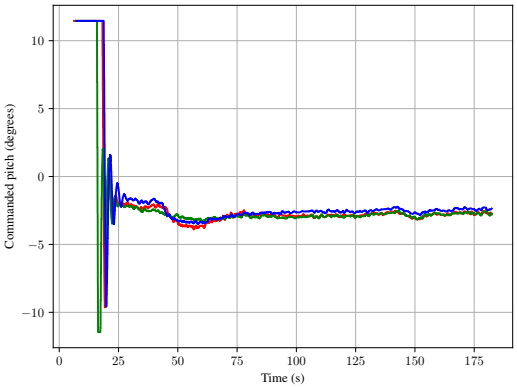

Figure 4.19: Commanded pitch.

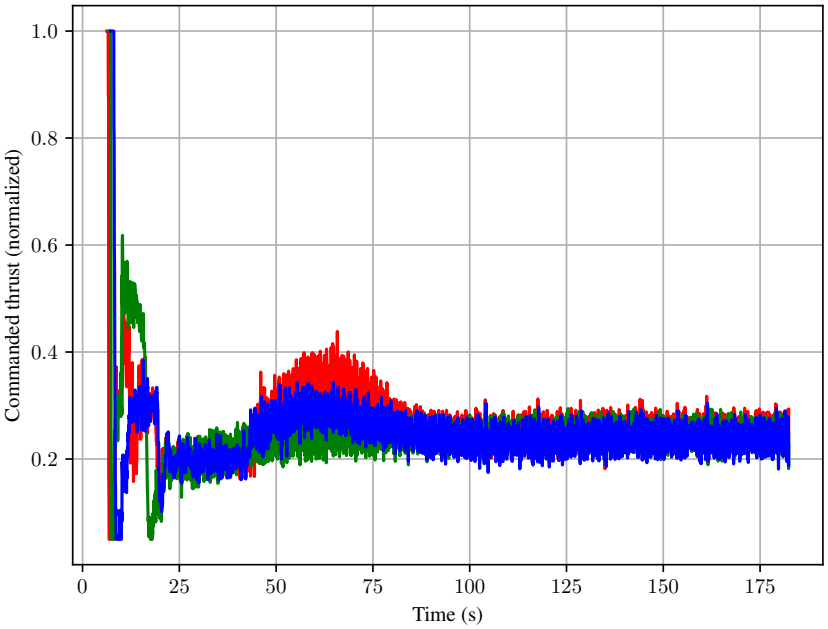

Figure 4.20: Thrust command.

Once again, the computed values were saturated before being sent to the autopilot's inner loop controller. The limit values were set to be the same defined in the numerical simulations, 10 and 20 meters per second for the commanded velocity and -0.4311 and 0.4311 radians (or approximately 25 degrees) for the commanded roll, as depicted in Figure 4.18. Limit values lower than 10 meters per second for the commanded velocity would result in an unstable velocity for the UAV to fly in, due to the lack of generated lift. The roll command limit values were chosen for similar reasons, to avoid instabilities and lower the chance of entering a stall. The remaining commands

to be sent directly to the autopilot are plotted in Figures 4.19 and 4.20. Figure 4.20 shows the thrust command over time, with limited commands between 0.05 and 1 (or 5% and 100% thrust power); the value stabilizes around 0.23, although with some low amplitude oscillations. The same behaviour can be observed in the plots depicting the roll and pitch commands, where there is a clear stabilization around the values of 8 and -3 degrees respectively, but also with an oscillatory component associated. The pitch commands is also limited to a certain maximum amplitude, this value being approximately 13 degrees.

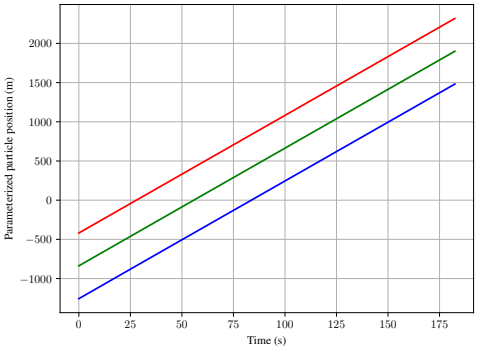

Figure 4.21: Parameterized particles positions, γ.

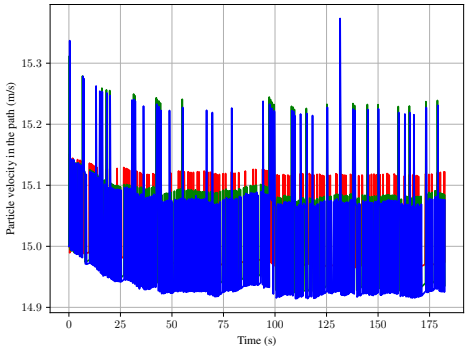

Figure 4.22: Particles velocities, $\dot{\gamma}$.

Through the analysis of the plotted data in Figure 4.21 and Figure 4.22, it is possible to conclude that the particles kept the desired distance between each other, maintaining a desired speed of around 15 meters per second, similar to the results obtained for the numerical simulations, although with low amplitude oscillations around the desired value.

The low amplitude oscillatory behaviour in the presented graphs (Figure 4.14, Figure 4.15, Figure 4.16, Figure 4.17, Figure 4.18, Figure 4.20 and Figure 4.22) could be related to the data transmission rate of the information packages through the control system. Another possible explanation could be the lack of signal filtering and adjusting. The overall behaviour is the desired one, with the commands converging to the necessary values to achieve the coordinated flight and the errors converging to a value around zero. One way to attenuate this behaviour would be to filter the particle velocities (shown in Figure 4.22) before sending the signal to the UAVs MPF controller. After that, exhaustive gain adjustment for the autopilot inner loop controller, PI controller and MPF (outer loop) controller could also improve the overall system performance. The addition of filters before sending the commanded signal to the aircraft autopilot's inner loop controller could possibly reduce this oscillations, but also undesirably increase delays in the error convergence to 0.

Since the aircrafts followed the assigned particles with an oscillating error that converges to a value around 0, while maintaining a constant desired altitude and the particles kept the distance between each other, at a constant desired speed, it is possible to validate the implementation for a coordinated flight around a steady target. The displayed views of the paths in the three dimensional plots give an intuition that the coordination is achieved, proved by the remaining analysed graphs presented. Based on the results shown, it is possible to validate the implementation, achieving the proposed solution for a coordinated multi-UAV flight around a steady target.

4.3.2 Moving Target

Similarly to what was done in Chapter 3, and taking into account that the ultimate objective is to follow a moving target, this section presents the results for a target with specified kinematics. In this case, the movement is defined as it was in the numerical simulations with the acceleration being given by $||\dot{\mathbf{v}}_t^I||(t) = 0.1\sin(0.07t)$ meters per second squared and the angular velocity governed by $\omega_t(t) = 0.02\cos(0.03t)$ radians per second. The presented results follow the same structure as the previous section for a steady target, omitting the particle coordination results since, as stated in Chapter 3, the results obtained are not changed by having a moving target.

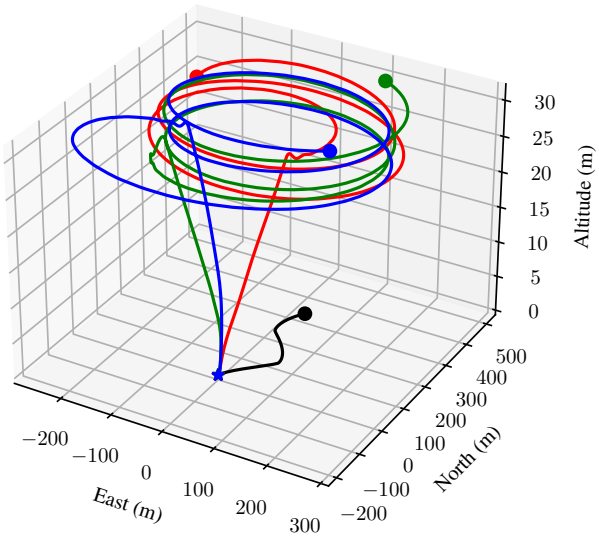

Figure 4.23: UAVs and target paths. The stars represent the UAVs starting positions and the circles the final positions.

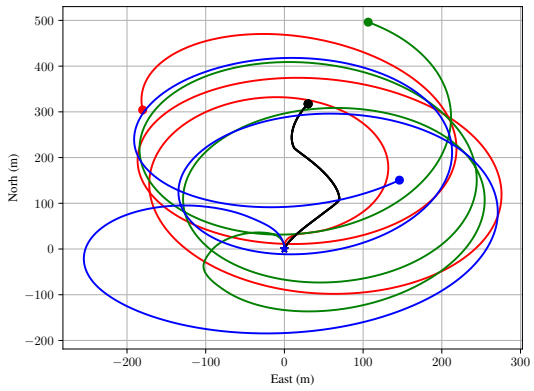

Figure 4.24: Longitudinal (North-East plane) perspective of the UAVs and target paths.

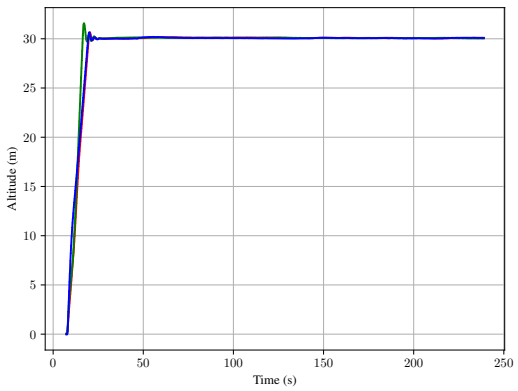

Figure 4.25: UAVs altitude (in meters) over time (in seconds).

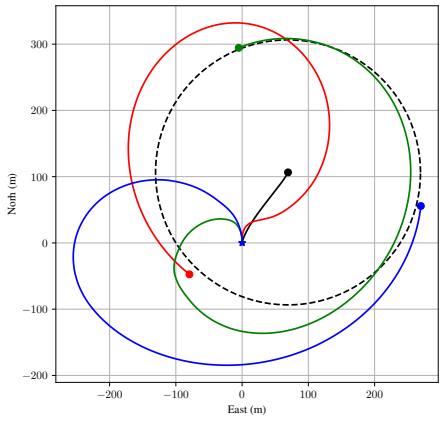

Figure 4.26: Longitudinal perspective at 75 seconds.

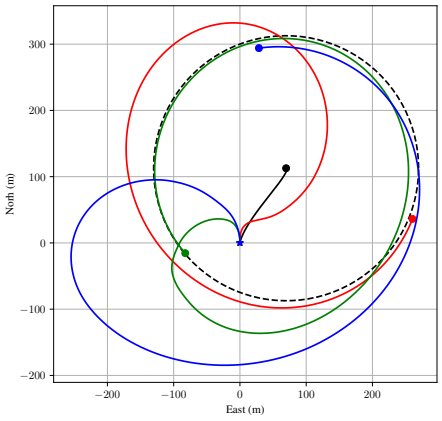

Figure 4.27: Longitudinal perspective at 100 seconds.

Figures 4.26 and 4.27 show the UAVs and target trajectories up until 75 and 100 seconds, respectively. The behaviour is similar to the one obtained for a steady target simulation, with the UAVs converging and maintaining themselves in the circular path, correctly following the target.

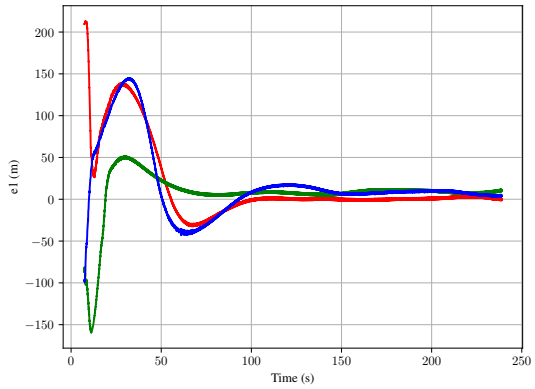

Figure 4.28: Error $\mathbf{e}_1$, as described in Section 3.2.1.

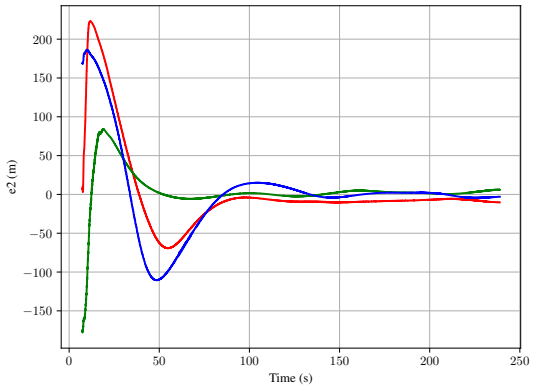

Figure 4.29: Error $\mathbf{e}_2$, as described in Section 3.2.1.

The errors plotted in Figures 4.28 and 4.29 converge to zero as time moves to infinity, as expected, similar to the results obtained in the numerical simulations and also in the previously shown simulation with a steady target.

Similar to what was obtained in the numerical simulations, the commanded values do not converge to a single value over time, since the aircraft needs to adapt its movement to follow correctly the circular path around the moving target. Once again, using the software in the loop, the oscillatory component is present in the computed command controls and in the tracking errors.

Analysing the plot in Figure 4.33 and comparing it with the previously obtained plot in Figure 4.19, it is possible to see that there are more variations in the commanded value for a moving target. This is associated with the changing commanded velocities, that influence the altitude at which the vehicles fly in. For instance, if the target decelerates, than the vehicle will also decelerate, meaning that its airspeed will be lower, which by itself,

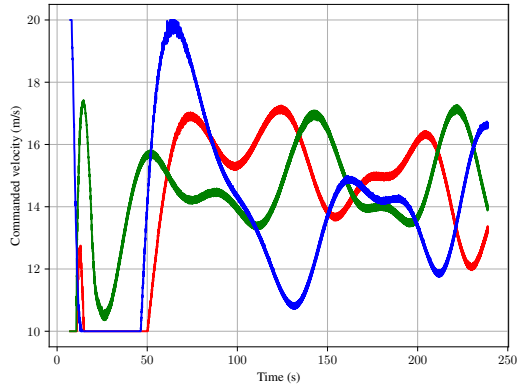

Figure 4.30: Commanded velocity.

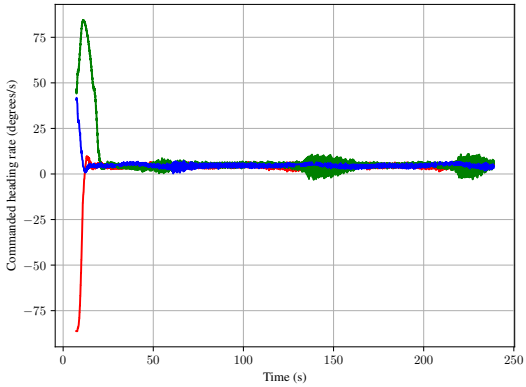

Figure 4.31: Commanded heading rate.

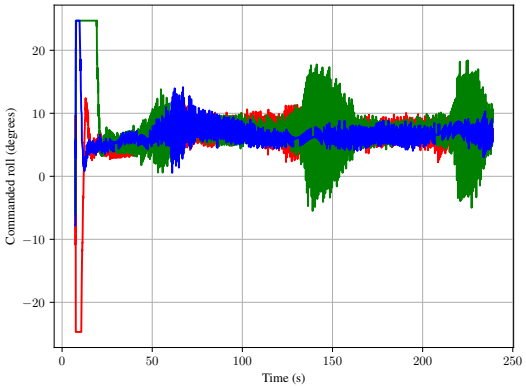

Figure 4.32: Commanded roll.

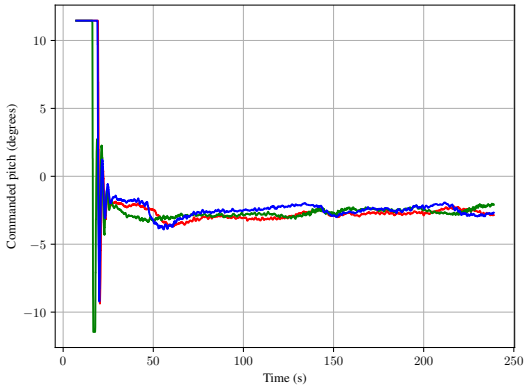

Figure 4.33: Commanded pitch.

decreases the UAV altitude. To remain at the same altitude, the aircraft must adjust its pitch, which explains the observed behaviour.

Taking into account the presented results, it is safe to assume that the implementation is validated with the errors converging to a value around zero and achieving a coordinated flight around a moving target, as desired. The next chapter explores the use of a computer vision system to obtain the target data.

4.4 Practical Considerations

The way the nodes are automatically instantiated for every aircraft with the execution of a single ".launch" file allows for an easy simulation deployment and, eventually, flight tests. Overall, the developed work increased the software setup quality, facilitating future projects and tests needed to move to real mission applicability.

With the implemented architecture, it is possible to easily choose different parameters in order to have, for instance, a different number of simulated aircrafts to perform the coordinated flight, different followed paths or non equal distances between vehicles. Depending on the mission requirements, having this abstraction capability for so many different flight characteristics presents is an advantage to CIAFA.

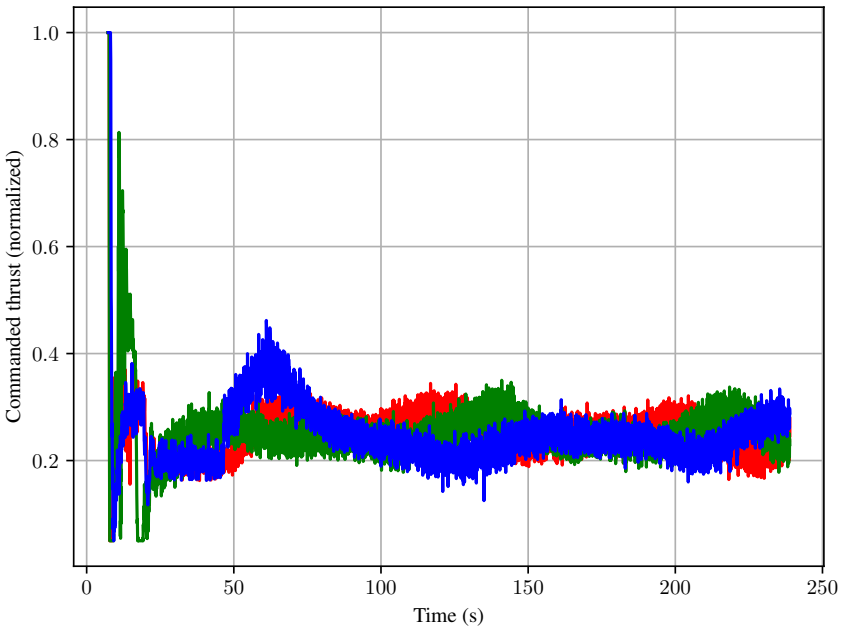

Figure 4.34: Thrust command.

There is, however, a limit for the number of vehicles deployed, given how the scalability was implemented. For every new UAV added to the simulation, there are new ports being defined. This ports consist in a sum of a basis number, that would be the number associated with a single vehicle simulation, and the vehicle identification number, which is defined in the ".launch" files, and it must be unique for every communication link. It is important that every vehicle has its own ports well defined so that there are no problems in the data transmission between the represented blocks in Figure 4.4. As it is implemented, the ports definition limits the maximum number to ten simulated aircrafts. Although the port assignment strategy is a limitation, it can be easily fixed by choosing a different port range in the future.

One possible solution for the static error would be to add the error integral term to the control commands. This error might be related with autopilot inner loop controller low efficiency to perform the desired maneuvers, due to inherited natural communication delays between the sent attitude command and the actual deflection of control surfaces.

The implemented architecture simulation setup is available in an online repository (Félix (2022)). All the files used and written code are well documented, and there are also instructions on how to setup the simulations in any machine with the correct configurations and necessary programs.

Chapter 5

Closed Loop Simulations with Computer Vision

In order to implement a more realistic and complete simulation in the loop, the target data should be acquired by an UAV onboard sensor. For this purpose, a simulated electro-optic sensor was used together with a target detection algorithm using computer vision. This section aims to explain the implementation and show its results, based on the work done in Alves et al. (2022).

5.1 Implementation

Recalling the scheme presented in Figure 4.5, the grey target node block depicted the node that published the information on the target movement to all the UAVs. As it was mentioned in the proposed solutions in Chapter 3, the objective is to implement a distributed command and control architecture. However, for the UAVs to obtain the target data, the node publishes this data in a centralized manner. The best solution for the architecture implementation in order to achieve a distributed design is outside the scope of this thesis and, being so, the implementation of a distributed architecture should be integrated in future works that explore how the data from all the UAVs can be fused together and then broadcast over the UAV network.

By having cameras mounted onto the UAVs, the target can be detected through computer vision algorithms based on the acquired camera images. To simply validate the proposed control system, which has a distributed architecture in terms of the trajectory coordination, a single UAV captures, processes and publishes the target coordinates and velocity to all the other formation UAVs, in a centralized manner.

The full implemented ROS environment can be found in Figure 5.1, where the target publishing data module is replaced by a new target detection based on computer vision algorithm used and developed in Alves et al. (2022), an "You Only Look Once" (Redmon, Divvala, Girshick, and Farhadi (2015)) detector type computer vision algorithm. The diagram displayed in the Figure 5.2 shows the target detecting algorithm with all its nodes and connections in detail.

Starting from bottom to top, in Figure 5.2, the first node is implemented in the image publisher block and it publishes the obtained camera image to a topic. This image is generated in the Gazebo simulation environment,

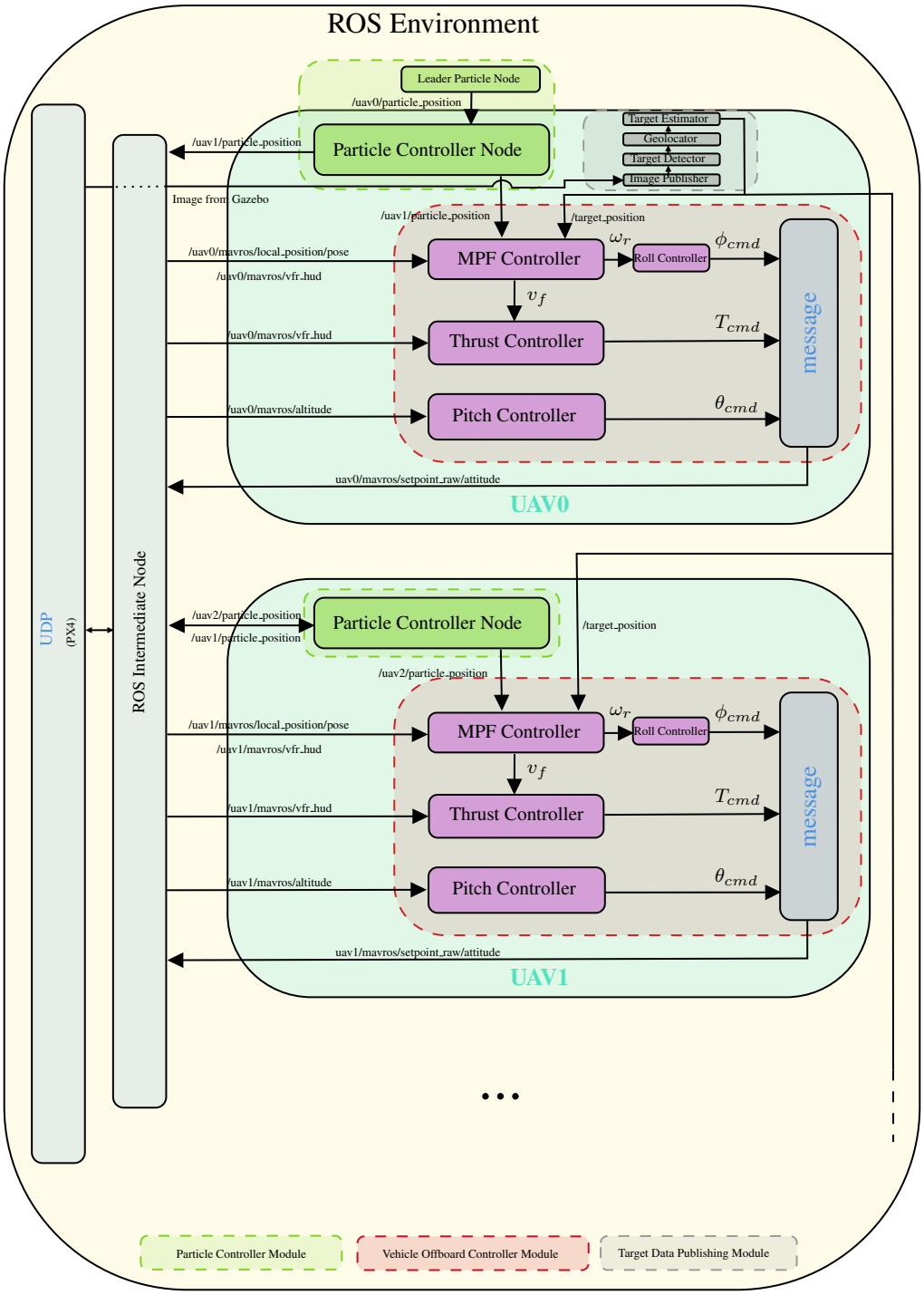

Figure 5.1: ROS Environment (taken from Figure 4.8) with target detecting algorithm implemented in the Target Data Publishing Module.

in which the first UAV is equipped with a transmitting camera that passes the image through an UDP port. The second node, represented in the target detector block, detects the target position in two dimensions. The next node, implemented in the geolocator block, translates the two dimensional position of the target in the image into a three dimensional position in relation to the inertial frame, which can then be used as an input to the control law. The target estimator node, implemented in the position estimator block estimates the target data through time even when there are no updates being published by the previous node, so that the control law receives the most

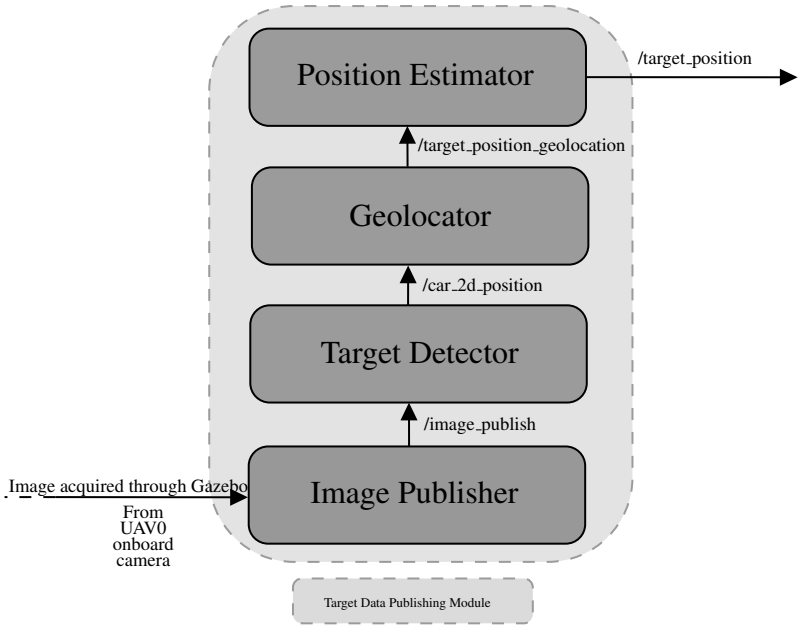

Figure 5.2: Detailed target detecting algorithm structure, from the Target Data Publishing Module in 5.1

reliable information on the target movement; this estimation process is done using a Kalman filter (Alves et al. (2022)). Finally, the estimator publishes the target estimations in a topic to be used by all the simulated vehicles. The target detection architecture was implemented in a sequential and modular way, which is an advantage for future changes or new implementations; for instance, the implemented target detector only detects red cars but this can be easily changed by replacing the target detector block with another different detecting algorithm, due to the adopted modular approach.

5.2 Results

In this new set of simulations, the circular path radius was defined to be 100 meters, in order to comply with the original implementation in Alves et al. (2022). This parameter directly affects the angle at which the camera will be pointing to the target, influencing the performance of the target detection algorithm.

A new object had to be spawned in the simulated Gazebo world, to serve as the target to be detected. In Alves et al. (2022), the algorithm was trained and validated for the detection of a red car, so a red car model was added into the world to simulate the target. The simulation conditions are the same as detailed in Chapter 4.

In the presented graphs, each colour represents, once again, a different aircraft. The black colour represents the estimated target position and the magenta colour represents the real target position.

5.2.1 Steady target

The paths drawn by the aircrafts in Figure 5.3 (in three dimensions), Figure 5.5 (longitudinal perspective), Figure 5.6 and Figure 5.7 are similar to the ones obtained in Subsection 4.3.1, with slightly more variation on the circles positions, centered on the target estimations given by the target detection algorithm. The target remained steady at the position $\mathbf{p}_I^t(t) = [0\ 0]$, but the algorithm estimated different positions through time, which is ex-

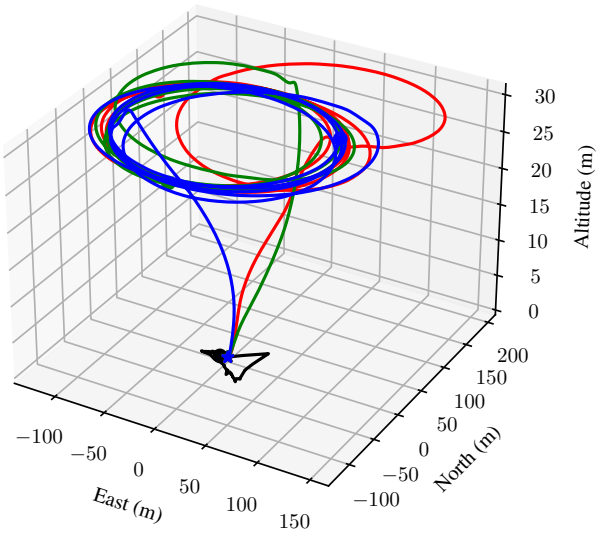

Figure 5.3: UAVs and target paths. The stars represent the UAVs starting positions and the circles the final positions.

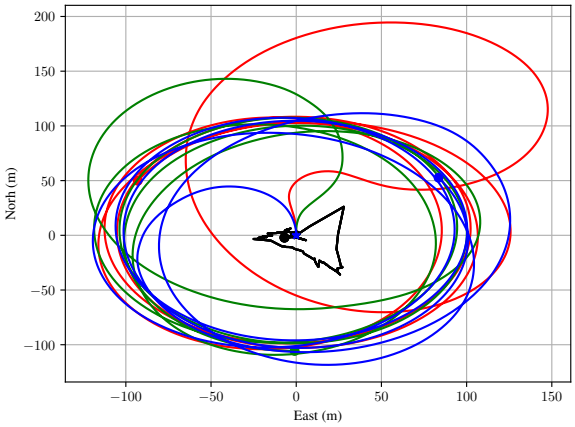

Figure 5.4: Longitudinal (North-East plane) perspective of the UAVs and target position estimations.

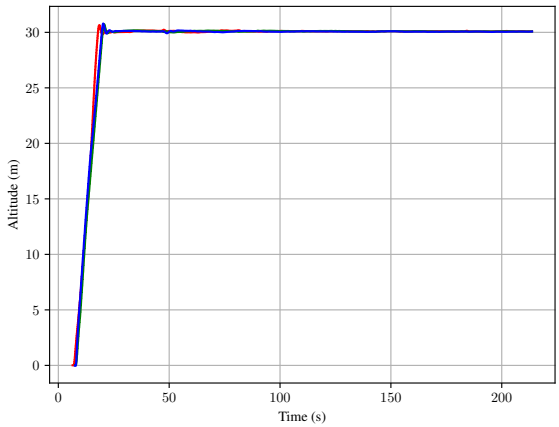

Figure 5.5: UAVs altitude (in meters) over time (in seconds).

pected given the results obtained in Alves et al. (2022). Overall, the coordinated flight is achieved around the real target position, while the first UAV is able to estimate the target position and velocity and provide this data to the remaining UAVs.

The plots presented in Figures 5.8 and 5.9 suggest that the error converges to a small value around zero as time moves to infinity. It is important to notice that there is an overall worst performance when compared with the results obtained in Subsection 4.3.1, taking more time to reach the same error amplitudes, which is expected given the bigger target position input variations derived from the target estimator algorithm. At approximately 180

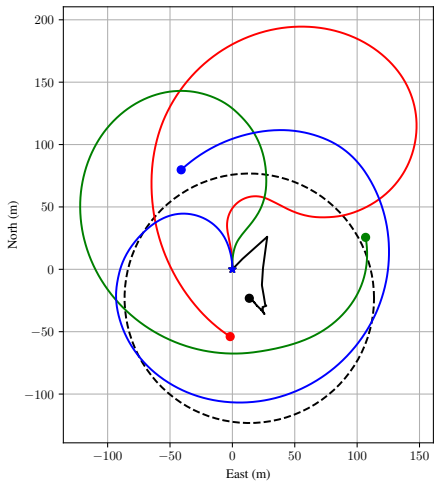

Figure 5.6: Longitudinal perspective at 50 seconds. The circles represent the last recorded positions.

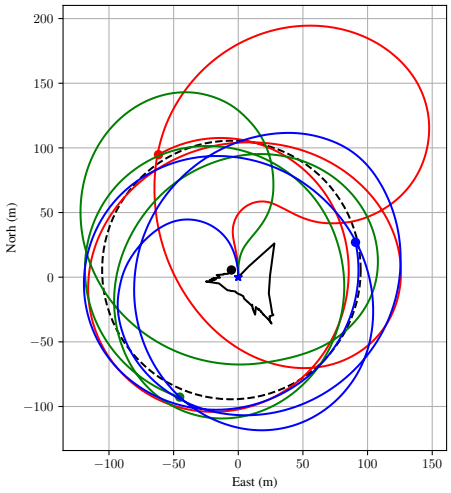

Figure 5.7: Longitudinal perspective at 120 seconds. The circles represent the last recorded position.

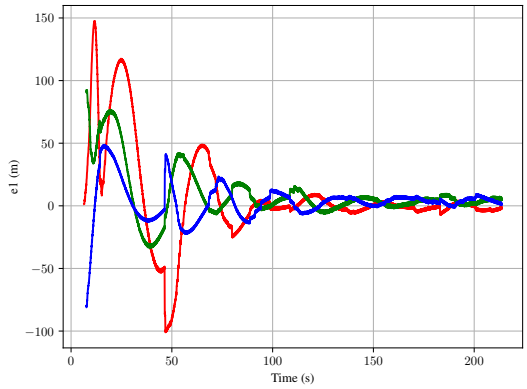

Figure 5.8: Error $\mathbf{e}_1$, as described in Section 3.2.1.

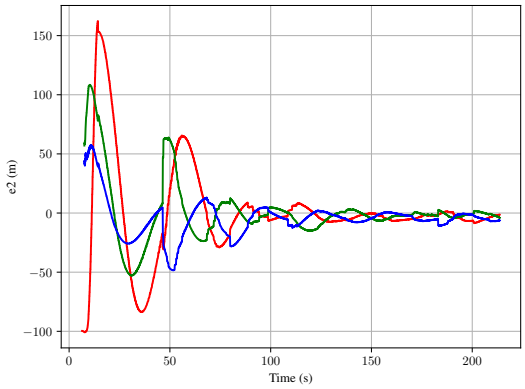

Figure 5.9: Error $\mathbf{e}_2$, as described in Section 3.2.1.

seconds (see Figures 5.8 and 5.9), there is a sudden increase in amplitude in both errors, derived from a possible sudden change in the target position estimate value fed to the control law. Despite this, the error still remains in an converging tendency towards 0. As long as the target estimates remain stable, the error should also stabilize, tending to 0.

Following the same logic as the previous chapters, now that the implementation is validated for a steady target, the next subsection explores the effects of this added feature for the coordinated flight around a moving target.

5.2.2 Moving target

In the previous simulations with a moving target, the values for its position and velocity were published based on the time passed and a set of equations that governed the target movement, as mentioned, for instance, in the beginning of Subsection 4.3.2. Since, in this setup, the target estimates are obtained from a target detector algorithm, it is necessary to move the car object in the Gazebo simulated world. In order to achieve that, a set of

waypoints were defined and the object was programmed to follow said waypoints, one at a time in a sequential way. The points chosen to govern the car movement are described in Table 5.1. This positions are represented with a magenta colour throughout the presented figures.

Target Waypoints			
Order	Time [s]	North [m]	East [m]
1	0	0	0
2	50	0	0
3	100	0	50
4	150	0	150
5	175	0	175
6	200	0	250
7	250	0	150
8	300	0	0

Table 5.1: Defined waypoints to govern target movement.

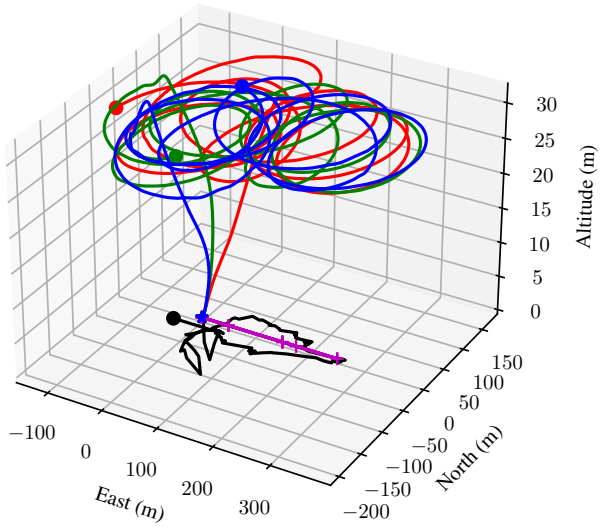

Figure 5.10: UAVs and target paths. The stars represent the UAVs starting positions and the circles the final positions. The target real position is depicted in magenta, where the magenta crosses represent the defined waypoints. The black line represents the computer vision system estimated target's position.

Based on the waypoints defined in Table 5.1, the expected black line in the plots in Figures 5.10 and 5.11 would be a straight line going in the East positive direction, keeping the North coordinate at 0 value. This is not the case, as it can be observed, since the black line presents a curved shape, indicating estimation errors compared to the real target position. From Figure 5.13 it is possible to visually conclude that the target estimations had a poor performance, with large estimation errors compared with the real target position. This is due to the fact that, during long periods of time, the UAV with the mounted camera could not capture the target on the acquired images. After the target is captured in the images and the UAVs start to correctly circling over it, the estimations start to get better and coordinated flight following the moving target is achieved. After 200 seconds, the target reaches its further

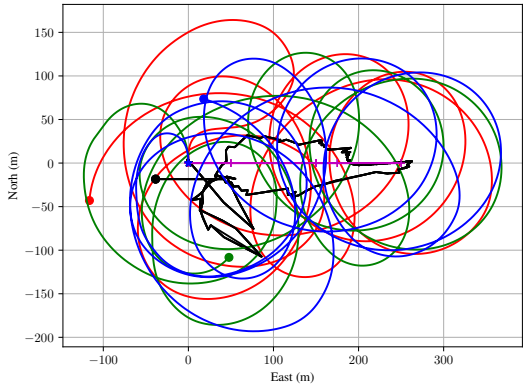

Figure 5.11: Longitudinal (North-East plane) perspective of the UAVs and target paths.

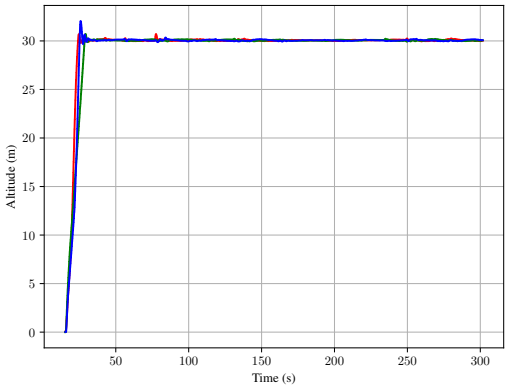

Figure 5.12: UAVs altitude (in meters) over time (in seconds).

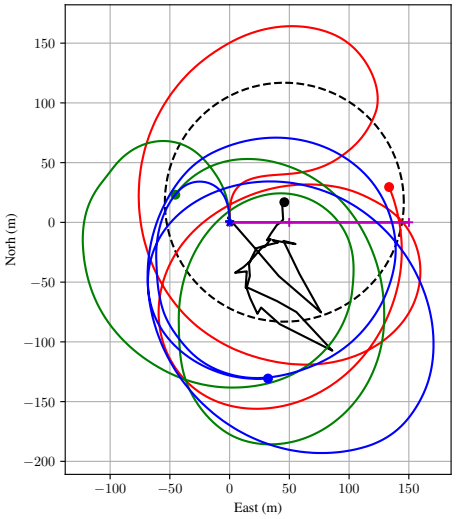

Figure 5.13: Longitudinal perspective at 150 seconds. The circles represent the last recorded positions.

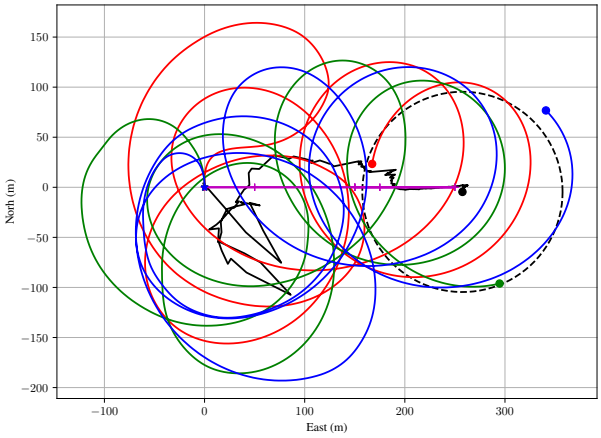

Figure 5.14: Longitudinal perspective at 200 seconds. The circles represent the last recorded positions.

most position and starts heading back to the origin, and its clear from Figure 5.14 that throughout this simulation period, the aircrafts followed in a coordinated way the moving target. Figure 5.11 show the UAVs full trajectory, where they headed back to the origin, correctly following once again the moving target.

Regarding the errors plotted in Figures 5.15 and 5.16, although there is no clear convergence to 0, there is a tendency for lower error amplitudes as time goes forward. After 200 seconds, the target changes its movement to the opposite direction, and there is an increase in the error amplitude during approximately 50 seconds, after which it starts to decrease again, when the estimations stabilize and the target estimation algorithm correctly identifies the target direction change.

Overall, the target could be followed in a coordinated way, proving the conjugation of an image fed target position and velocity estimator with the proposed architecture. The results validate the implementation and the

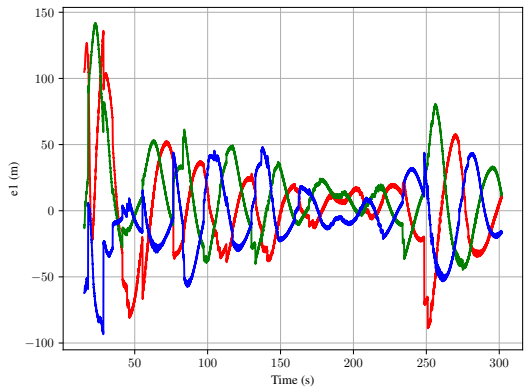

Figure 5.15: Error $\mathbf{e}_1$, as described in Section 3.2.1.

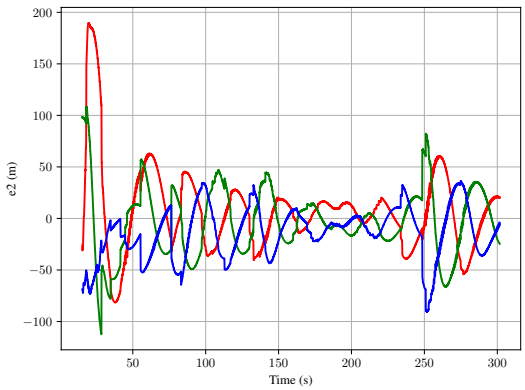

Figure 5.16: Error $\mathbf{e}_2$, as described in Section 3.2.1.

ultimate goal of following a moving target in a coordinated manner is once again achieved, this time with the added capability of real time estimation based on the acquired images from a simulated electro-optic sensor.

5.3 Practical Considerations

In order to program the target movement, several waypoints need to be manually defined and coded into the object spawning file. There are ways to implement the governing equations as done in previous chapters, namely by implementing a plugin into the Gazebo simulations environment. It includes programming in C++ language and integrating the file into the Gazebo plugins folder. (Gazebo (2022))

It is important to have a feature to filter out the detection of other vehicles that take part in the mission. If there is a detection of an UAV that is part of the flight formation, this detection should be correctly identified as such and discarded. Since the implementation done in Alves et al. (2022) only focused on validating the algorithm for a single aircraft, this filter is not yet integrated and still needs to be added to the overall target detection algorithm.

It is necessary that for most of the simulation time, the target is inside the captured image frame. If there are long periods of time with no target detections, the UAV will estimate wrong values, which by itself will make the UAVs drift from the target actual position, generating unwanted cumulative errors. It is recommended that in the future architecture developments, the aircrafts switch to searching mode after some defined time without detections.

Chapter 6

Conclusions

Regarding all the work developed in this thesis, it is important to highlight the main ideas and achievements, discuss the results obtained and outline complementary and future projects to be developed.

6.1 Conclusions

The developed work aimed to achieve the formal implementation of a command and control architecture to be used in search and target tracking missions in a maritime environment. Based on the idealized architecture in Santos (2021), this thesis specifically contributes to the implementation of the target tracking mode of that architecture. The proposed implementation allows for a set of UAVs to follow in a coordinated way a circular moving path centered around a moving target, using the MPF method (Oliveira, Aguiar, and Encarnacao (2016) and Jain (2019)).

After a thorough review of the available literature, several solutions are considered. The proposed solutions to answer the problems stated at the beginning of Chapter 1 were presented and their choice was justified. For the UAVs controllers, the MPF method was implemented, since it has smooth convergence to the moving path and it was successfully tested and validated in practical applications with CIAFA's UAVs. A circular path was designed for the flight formation of the fixed-wing UAVs, in which they are equally spaced and always moving along the path. A distributed command and control architecture was implemented, with data shared between UAVs in a chained network, allowing for modular coordination control and future implementations using shared data to better estimate the target, for instance.

Regarding the results obtained in the numerical simulations from Chapter 3, it is safe to say that in all situations (whether with a steady or moving target), the implemented proposed control law successfully commanded the system. Overall, the system is well behaved with a clear exponential decay tending to 0 in all the error plots presented. The resulting commanded control inputs converged to the expected values, both to steer the vehicle in the right direction and to control the particle to coordinate the formation.

When implementing a more realistic environment in Chapter 4, a low amplitude oscillatory behaviour can be seen in the error plots, together with the exponential decay tendency to zero. The exponential aspect of the error evolution is the expected one, since the outer control loop was validated in the previous chapter with the same

results. The new oscillatory behaviour is most likely related to the lack of signal filtering and gain adjustment. This could be possibly solved, for instance, by applying a filter to the particles commanded velocity, although it was not implemented in this dissertation.

The addition of a sensor to obtain the data on the target in Chapter 5 added a more practical and realistic component to the proposed solution. Through the use of computer vision algorithms, the target data could be correctly estimated. The results obtained proved the successful implementation, with slightly worse performance when compared with having the exact target location and velocity. Although the target detector only works when the target is a red car, due to the modular implementation, the detection algorithm can be easily changed to detect whatever target it is trained for, using different sensors. The results also show problems may occur when, during long periods of time, the target is not captured in the acquired images from the UAV onboard electro-optic sensor.

Since all the work developed followed a modular and incremental approach, it is possible to change some of the algorithms and code in a simple way by replacing the blocks depicted in Figure 4.8, as it was done in the last section of Chapter 4 with the target data block. This allowed for the main overall structure to remain unaffected by the replacement of the individual modular parts. This approach is advantageous since it allows for future projects to easily implement their ideas and also to pinpoint where the errors are during the implementation phase.

Also, the implementation allows for easy scalability and changeable and addable complementary features. Parameters can easily be changed in order to achieve the best performance and desired results for a given scenario.

The creation of online repositories may serve as a foundation for future work to be developed. These repositories are documented and updated with the results presented in this dissertation. The complete simulation tool-chain can be easily setup in any machine with the correct configurations and necessary programs installed. This setup contemplates the most updated software available, which increases CIAFA's software setup quality.

In the end, a distributed control architecture was implemented with satisfactory results. With the proposed method and implementation, a parameterized number of aircrafts can follow a circular path around a generally moving target, while keeping a maximum distance between each other, as proved by the results presented throughout this dissertation.

6.2 Future Work

Several projects can be developed in the future to complement and improve the present work in this dissertation. There is always room for improvement and optimization, new techniques to explore and novel and creative ideas to implement. In the following paragraphs, some ideas are discussed regarding future work.

Starting with the integration of this module in the overall architecture proposed in Santos (2021), there is still objectives to pursue. Namely, having all three modes implemented in a single architecture: target searching, target tracking and collision avoidance. All the different scenarios should be investigated to assure flight safety and the viability of the implementation as a whole. While this work addresses the target tracking mode, the transition between flight modes and its corresponding transition conditions remains to be studied in depth.

The design and integration of a collision avoidance algorithm in the architecture is an essential piece for the project as an whole, since it improves flight safety and assures a robust mission accomplishment. This problem could be approached in two main different ways: design a completely new control law that switches when the

aircrafts get to close to each other or use the previously implemented control laws and try to conjugate them with this desired capability.

Regarding the target detection implementation, a lot of work can still be developed. Future projects could focus on gimbaled camera systems to detect and always point at the target while moving around it or even use the captured images to detect other aircrafts or obstacles, using that information to feed a collision avoidance algorithm. If there is more than one aircraft being used, the acquired images from each one of them could be used to feed a sensor fusion algorithm to better estimate the target, for instance. The data sharing architecture or the way that the estimation is done using all the images together are both interesting and useful fields of research.

Although well documented and explicit, so that a new user can easily understand how everything is implemented, there are several improvements that could be added to the simulated architecture. For instance, add functionalities to detect whether the needed files, libraries or packages are accessible or not, implement a try and except logic in instantiations or change the Python files that start the nodes to be defined as classes. There is also a scalability problem related to the definition of the ports that limits the number of vehicles to 10, which can be a problem for some specific applications where a large number of UAVs is required.

Finally, another potentially interesting topic to explore would be to investigate different control laws, particularly to control the particle dynamics. Desirably, the particle velocity could be adjusted depending on the target velocity, guaranteeing that the desired UAV commanded velocity is kept within its flight envelope.

Bibliography

Abdelkader, M., Güler, S., Jaleel, H., & Shamma, J. S. (2021, July). Aerial swarms: Recent applications and challenges. *Current Robotics Reports*, *2*(3), 309–320. Retrieved from https://doi.org/10.1007/s43154-021-00063-4 doi: 10.1007/s43154-021-00063-4

AFA. (2022). *Portuguese Air Force Academy website.* From AFA - Academia da Força Aérea Portuguesa. Retrieved from https://www.academiafa.edu.pt/

Aguiar, A. P., & Hespanha, J. P. (2007, August). Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty. *IEEE Transactions on Automatic Control*, *52*(8), 1362–1379. Retrieved from https://doi.org/10.1109/tac.2007.902731 doi: 10.1109/tac.2007.902731

Ahmed, M., & Subbarao, K. (2010, December). Nonlinear 3-D trajectory guidance for unmanned aerial vehicles. In *2010 11th international conference on control automation robotics & vision.* IEEE. Retrieved from https://doi.org/10.1109/icarcv.2010.5707911 doi: 10.1109/icarcv.2010.5707911

Alves, J., Oliveira, T., Cruz, G., & Silva, D. (2022). *Software architecture for low-cost UAVs: an application considering automatic target tracking mission scenarios.* EasyChair Preprint no. 9143.

Armanini, S. F., Caetano, J. V., de Croon, G. C. H. E., de Visser, C. C., & Mulder, M. (2016, June). Quasi-steady aerodynamic model of clap-and-fling flapping MAV and validation using free-flight data. *Bioinspiration & Biomimetics*, *11*(4), 046002. Retrieved from https://doi.org/10.1088/1748-3190/11/4/046002 doi: 10.1088/1748-3190/11/4/046002

Arrichiello, F., Chiaverini, S., & Fossen, T. (2006). Formation control of marine surface vessels using the null-space-based behavioral control. In *Group coordination and cooperative control* (pp. 1–19). Springer-Verlag. Retrieved from https://doi.org/10.1007/11505532_1 doi: 10.1007/11505532_1

Bella, S., Belbachir, A., & Belalem, G. (2019). A centralized architecture for cooperative air-sea vehicles using UAV-USV. *International Journal of Computer and Information Engineering*, *13*(4), 201 - 210. Retrieved from https://publications.waset.org/vol/148

Bernard, M., Kondak, K., Maza, I., & Ollero, A. (2011, October). Autonomous transportation and deployment with aerial robots for search and rescue missions. *Journal of Field Robotics*, *28*(6), 914–931. Retrieved from https://doi.org/10.1002/rob.20401 doi: 10.1002/rob.20401

Brahim, T. (2018, 03). Laser beam steering along 3d paths. *IEEE/ASME Transactions on Mechatronics*, *PP*. doi: 10.1109/TMECH.2018.2821239

Breivik, M., Hovstein, V., & Fossen, T. (2008, 10). Straight-line target tracking for unmanned surface vehicles.

Modeling, Identification and Control (MIC), 131-149. doi: 10.4173/mic.2008.4.2

Briñón-Arranz, L., Seuret, A., & Pascoal, A. (2017, July). Target tracking via a circular formation of unicycles. *IFAC-PapersOnLine*, *50*(1), 5782–5787. Retrieved from https://doi.org/10.1016/j.ifacol.2017.08.422 doi: 10.1016/j.ifacol.2017.08.422

Briñón-Arranz, L., Seuret, A., & Pascoal, A. (2019, March). Circular formation control for cooperative target tracking with limited information. *Journal of the Franklin Institute*, *356*(4), 1771–1788. Retrieved from https://doi.org/10.1016/j.jfranklin.2018.12.011 doi: 10.1016/j.jfranklin.2018.12.011

Brust, M. R., Zurad, M., Hentges, L., Gomes, L., Danoy, G., & Bouvry, P. (2017, June). Target tracking optimization of UAV swarms based on dual-pheromone clustering. In *2017 3rd IEEE international conference on cybernetics (CYBCONF)*. IEEE. Retrieved from https://doi.org/10.1109/cybconf.2017.7985815 doi: 10.1109/cybconf.2017.7985815

Cao, C., Hovakimyan, N., Patel, V., Kaminer, I., & Dobrokhodov, V. (2007, July). Stabilization of cascaded systems via l1 adaptive controller with application to a UAV path following problem and flight test results. In *2007 american control conference*. IEEE. Retrieved from https://doi.org/10.1109/acc.2007.4283028 doi: 10.1109/acc.2007.4283028

Chitrakaran, V. K., Dawson, D. M., Kannan, H., & Feemster, M. (2006). Vision assisted autonomous path following for unmanned aerial vehicles. In *Proceedings of the 45th IEEE conference on decision and control*. IEEE. Retrieved from https://doi.org/10.1109/cdc.2006.377305 doi: 10.1109/cdc.2006.377305

Cruz, G., & Bernardino, A. (2015). Image saliency applied to infrared images for unmanned maritime monitoring. In *Lecture notes in computer science* (pp. 511–522). Springer International Publishing. Retrieved from https://doi.org/10.1007/978-3-319-20904-3_46 doi: 10.1007/978-3-319-20904-3_46

Cunha, R., Silvestre, C., & Pascoal, A. (2003, September). A path following controller for model-scale helicopters. In *2003 european control conference (ECC)*. IEEE. Retrieved from https://doi.org/10.23919/ecc.2003.7085301 doi: 10.23919/ecc.2003.7085301

Dai, S.-L., He, S., Lin, H., & Wang, C. (2018, May). Platoon formation control with prescribed performance guarantees for USVs. *IEEE Transactions on Industrial Electronics*, *65*(5), 4237–4246. Retrieved from https://doi.org/10.1109/tie.2017.2758743 doi: 10.1109/tie.2017.2758743

de Wit, C. C., Khennouf, H., Samson, C., & Sordalen, O. J. (1994, January). NONLINEAR CONTROL DESIGN FOR MOBILE ROBOTS. In *Recent trends in mobile robots* (pp. 121–156). WORLD SCIENTIFIC. Retrieved from https://doi.org/10.1142/9789814354301_0005 doi: 10.1142/9789814354301_0005

Elston, J., Frew, E. W., & Member, I. (2008). *E.: Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks.*

Encarnacao, P., & Pascoal, A. (2000). 3d path following for autonomous underwater vehicle. In *Proceedings of the 39th IEEE conference on decision and control (cat. no.00ch37187)*. IEEE. Retrieved from https://doi.org/10.1109/cdc.2000.914272 doi: 10.1109/cdc.2000.914272

Fahimi, F. (2007, June). Sliding-mode formation control for underactuated surface vessels. *IEEE Transactions on Robotics*, *23*(3), 617–622. Retrieved from https://doi.org/10.1109/tro.2007.898961 doi: 10.1109/tro.2007.898961

FAP. (2022). *Portuguese Air Force website.* From Força Aérea Portuguesa. Retrieved from https://www.emfa

.pt/esquadra-153-esquadra-991-8211-harpias

Felix, L. F., Gomes, A., & Suleman, A. (2013, April). Topology optimization of a wing including self-weight load. In *54th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics, and materials conference.* American Institute of Aeronautics and Astronautics. Retrieved from https://doi.org/10.2514/6.2013-1869 doi: 10.2514/6.2013-1869

Frew, E. W., Lawrence, D. A., Dixon, C., Elston, J., & Pisano, W. J. (2007, July). Lyapunov guidance vector fields for unmanned aircraft applications. In *2007 american control conference.* IEEE. Retrieved from https://doi.org/10.1109/acc.2007.4282974 doi: 10.1109/acc.2007.4282974

Félix. (2022). *Software in the loop simulations repository.* From Github. Retrieved from https://github.com/mg-felix/sitl-simulations

Gancet, J., Hattenberger, G., Alami, R., & Lacroix, S. (2005). Task planning and control for a multi-UAV system: architecture and algorithms. In *2005 ieee/rsj international conference on intelligent robots and systems* (p. 1017-1022). doi: 10.1109/IROS.2005.1545217

Gazebo. (2022). *Gazebo website.* From Gazebo Tutorials. Retrieved from https://classic.gazebosim.org/tutorials?tut=plugins_model

Goerzen, C., Kong, Z., & Mettler, B. (2009, November). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, *57*(1-4), 65–100. Retrieved from https://doi.org/10.1007/s10846-009-9383-1 doi: 10.1007/s10846-009-9383-1

Gonçalves, P., Sobral, J., & Ferreira, L. (2017, January). Establishment of an initial maintenance program for UAVs based on reliability principles. *Aircraft Engineering and Aerospace Technology*, *89*(1), 66–75. Retrieved from https://doi.org/10.1108/aeat-09-2014-0146 doi: 10.1108/aeat-09-2014-0146

Gonçalves, P., Sobral, J., & Ferreira, L. (2016). Reliability database for unmanned aerial vehicles based on morphological analysis. *The Aeronautical Journal*, *120*(1230), 1262–1274. doi: 10.1017/aer.2016.56

Guan, Z., Liu, H., Zheng, Z., Ma, Y., & Zhu, T. (2022). Moving path following with integrated direct lift control for carrier landing. *Aerospace Science and Technology*, *120*, 107247. Retrieved from https://www.sciencedirect.com/science/article/pii/S1270963821007574 doi: https://doi.org/10.1016/j.ast.2021.107247

Hamilton, T., & Ochmanek, D. (2020). *Operating low-cost, reusable unmanned aerial vehicles in contested environments: Preliminary evaluation of operational concepts*. RAND Corporation. Retrieved from https://doi.org/10.7249/rr4407 doi: 10.7249/rr4407

Han, J., hong Wang, C., & xing Yi, G. (2013, June). Cooperative control of UAV based on multi-agent system. In *2013 IEEE 8th conference on industrial electronics and applications (ICIEA).* IEEE. Retrieved from https://doi.org/10.1109/iciea.2013.6566347 doi: 10.1109/iciea.2013.6566347

Healey, A., & Lienard, D. (1993, July). Multivariable sliding mode control for autonomous diving and steering of unmanned underwater vehicles. *IEEE Journal of Oceanic Engineering*, *18*(3), 327–339. Retrieved from https://doi.org/10.1109/joe.1993.236372 doi: 10.1109/joe.1993.236372

Hejase, M., Noura, H., & Drak, A. (2015, 01). Formation flight of small scale unmanned aerial vehicles: A review. In (p. 221-248). doi: 10.1002/9781634827300ch10

International Civil Aviation Organization, I. M. O. (2016). Iamsar manual - international aeronautical and maritime

search and rescue manual (10th ed.) [Computer software manual]. London.

Jain, R. P. K. (2019). *Decentralized cooperative control methods for multiple mobile robotic vehicles* (Unpublished master's thesis). Faculdade de Engenharia da Universidade do Porto.

Kaminer, I., Pascoal, A., Xargay, E., Hovakimyan, N., Cao, C., & Dobrokhodov, V. (2010, March). Path following for small unmanned aerial vehicles using l1 adaptive augmentation of commercial autopilots. *Journal of Guidance, Control, and Dynamics*, *33*(2), 550–564. Retrieved from https://doi.org/10.2514/1.42056 doi: 10.2514/1.42056

Kaminer, I., Yakimenko, O., Dobrokhodov, V., Pascoal, A., Hovakimyan, N., Cao, C., ... Patel, V. (2007). *Coordinated path following for time-critical missions of multiple UAVs via l1 adaptive output feedback controllers.* Calhoun. Retrieved from https://calhoun.nps.edu/handle/10945/50330

Khatib, O. (1985). Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 ieee international conference on robotics and automation* (Vol. 2, p. 500-505). doi: 10.1109/ROBOT.1985.1087247

Koenig, N., & Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 ieee/rsj international conference on intelligent robots and systems (iros) (ieee cat. no.04ch37566)* (Vol. 3, p. 2149-2154 vol.3). doi: 10.1109/IROS.2004.1389727

LambDrive. (2022). *Lambdrive website.* From Ground Control Station/Software for Pixhawk Family of Flight Controllers. Retrieved from https://www.lambdrive.com/depot/Robotics/Controller/Ground-Control-Station/

Lee, S., Cho, A., & Kee, C. (2010, September). Integrated waypoint path generation and following of an unmanned aerial vehicle. *Aircraft Engineering and Aerospace Technology*, *82*(5), 296–304. Retrieved from https://doi.org/10.1108/00022661011092947 doi: 10.1108/00022661011092947

Leith, D. J., & Leithead, W. E. (2000, January). Survey of gain-scheduling analysis and design. *International Journal of Control*, *73*(11), 1001–1025. Retrieved from https://doi.org/10.1080/002071700411304 doi: 10.1080/002071700411304

Leonard, N., & Fiorelli, E. (2001). Virtual leaders, artificial potentials and coordinated control of groups. In *Proceedings of the 40th ieee conference on decision and control (cat. no.01ch37228)* (Vol. 3, p. 2968-2973 vol.3). doi: 10.1109/CDC.2001.980728

Li, Z., Sun, J., & Oh, S. (2010, June). Handling roll constraints for path following of marine surface vessels using coordinated rudder and propulsion control. In *Proceedings of the 2010 american control conference.* IEEE. Retrieved from https://doi.org/10.1109/acc.2010.5531275 doi: 10.1109/acc.2010.5531275

Liu, L., Wang, D., Peng, Z., Chen, C. L. P., & Li, T. (2019, April). Bounded neural network control for target tracking of underactuated autonomous surface vehicles in the presence of uncertain target dynamics. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(4), 1241–1249. Retrieved from https://doi.org/10.1109/tnnls.2018.2868978 doi: 10.1109/tnnls.2018.2868978

Mansouri, S. S., Nikolakopoulos, G., & Gustafsson, T. (2015, November). Distributed model predictive control for unmanned aerial vehicles. In *2015 workshop on research, education and development of unmanned aerial systems (RED-UAS).* IEEE. Retrieved from https://doi.org/10.1109/red-uas.2015.7441002 doi: 10.1109/red-uas.2015.7441002

Marques, J. S., Bernardino, A., Cruz, G., & Bento, M. (2014a). An algorithm for the detection of vessels in aerial images. In *2014 11th ieee international conference on advanced video and signal based surveillance (avss)* (p. 295-300). doi: 10.1109/AVSS.2014.6918684

Marques, J. S., Bernardino, A., Cruz, G., & Bento, M. (2014b, August). An algorithm for the detection of vessels in aerial images. In *2014 11th IEEE international conference on advanced video and signal based surveillance (AVSS)*. IEEE. Retrieved from https://doi.org/10.1109/avss.2014.6918684 doi: 10.1109/avss.2014.6918684

MathWorks. (2022a). *MATLAB website.* From MATLAB - MathWorks - MATLAB. Retrieved from https://www.mathworks.com/products/matlab.html

MathWorks. (2022b). *Simulink website.* From Simulink - Simulation and Model-Based Design - MATLAB & Simulink. Retrieved from https://www.mathworks.com/products/matlab.html

Maza, I., Kondak, K., Bernard, M., & Ollero, A. (2009). Multi-UAV cooperation and control for load transportation and deployment. In *Selected papers from the 2nd international symposium on UAVs, reno, nevada, u.s.a. june 8–10, 2009* (pp. 417–449). Springer Netherlands. Retrieved from https://doi.org/10.1007/978-90-481-8764-5_22 doi: 10.1007/978-90-481-8764-5_22

Maza, I., Ollero, A., Casado, E., & Scarlatti, D. (2015). Classification of multi-UAV architectures. In (chap. 38). Springer Science Business Media Dordrecht.

McLean, D. (1990). *Automatic flight control systems*. Prentice Hall. Retrieved from https://books.google.pt/books?id=cJNTAAAAMAAJ

Meier, L., Tanskanen, P., Fraundorfer, F., & Pollefeys, M. (2011). Pixhawk: A system for autonomous flight using onboard computer vision. In *2011 ieee international conference on robotics and automation* (p. 2992-2997). doi: 10.1109/ICRA.2011.5980229

Milhim, A., Zhang, Y., & Rabbath, C.-A. (2010, April). Gain scheduling based PID controller for fault tolerant control of quad-rotor UAV. In *AIAA infotech@aerospace 2010.* American Institute of Aeronautics and Astronautics. Retrieved from https://doi.org/10.2514/6.2010-3530 doi: 10.2514/6.2010-3530

Murray, R. M. (2007, May). Recent research in cooperative control of multivehicle systems. *Journal of Dynamic Systems, Measurement, and Control*, *129*(5), 571–583. Retrieved from https://doi.org/10.1115/1.2766721 doi: 10.1115/1.2766721

Nelson, D., Barber, D., McLain, T., & Beard, R. (2007, June). Vector field path following for miniature air vehicles. *IEEE Transactions on Robotics*, *23*(3), 519–529. Retrieved from https://doi.org/10.1109/tro.2007.898976 doi: 10.1109/tro.2007.898976

Nigam, N., Bieniawski, S., Kroo, I., & Vian, J. (2012, September). Control of multiple UAVs for persistent surveillance: Algorithm and flight test results. *IEEE Transactions on Control Systems Technology*, *20*(5), 1236–1251. Retrieved from https://doi.org/10.1109/tcst.2011.2167331 doi: 10.1109/tcst.2011.2167331

Oliveira, T., Aguiar, A. P., & Encarnacao, P. (2016, October). Moving path following for unmanned aerial vehicles with applications to single and multiple target tracking problems. *IEEE Transactions on Robotics*, *32*(5), 1062–1078. Retrieved from https://doi.org/10.1109/tro.2016.2593044 doi: 10.1109/tro.2016.2593044

Oliveira, T., Aguiar, A. P., & Encarnacao, P. (2017, May). Three dimensional moving path following for fixed-wing unmanned aerial vehicles. In *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE. Retrieved from https://doi.org/10.1109/icra.2017.7989315 doi: 10.1109/icra.2017.7989315

Oliveira, T., Aguiar, A. P., & Encarnação, P. (2016, June). A convoy protection strategy using the moving path following method. In *2016 international conference on unmanned aircraft systems (ICUAS)*. IEEE. Retrieved from https://doi.org/10.1109/icuas.2016.7502567 doi: 10.1109/icuas.2016.7502567

OpenCV. (2022). *Opencv website.* From OpenCV. Retrieved from https://opencv.org/

Parker, L. (1998, April). ALLIANCE: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, *14*(2), 220–240. Retrieved from https://doi.org/10.1109/70.681242 doi: 10.1109/70.681242

Paul, T., Krogstad, T. R., & Gravdahl, J. T. (2008, October). Modelling of UAV formation flight using 3d potential field. *Simulation Modelling Practice and Theory*, *16*(9), 1453–1462. Retrieved from https://doi.org/10.1016/j.simpat.2008.08.005 doi: 10.1016/j.simpat.2008.08.005

Peng, Z., Wang, D., Li, T., & Han, M. (2020, June). Output-feedback cooperative formation maneuvering of autonomous surface vehicles with connectivity preservation and collision avoidance. *IEEE Transactions on Cybernetics*, *50*(6), 2527–2535. Retrieved from https://doi.org/10.1109/tcyb.2019.2914717 doi: 10.1109/tcyb.2019.2914717

Peng, Z., Wang, J., & Wang, D. (2017, April). Containment maneuvering of marine surface vehicles with multiple parameterized paths via spatial-temporal decoupling. *IEEE/ASME Transactions on Mechatronics*, *22*(2), 1026–1036. Retrieved from https://doi.org/10.1109/tmech.2016.2632304 doi: 10.1109/tmech.2016.2632304

Peng, Z., Wang, J., & Wang, D. (2018, May). Distributed maneuvering of autonomous surface vehicles based on neurodynamic optimization and fuzzy approximation. *IEEE Transactions on Control Systems Technology*, *26*(3), 1083–1090. Retrieved from https://doi.org/10.1109/tcst.2017.2699167 doi: 10.1109/tcst.2017.2699167

Peng, Z., Wang, J., Wang, D., & Han, Q.-L. (2021, February). An overview of recent advances in coordinated control of multiple autonomous surface vehicles. *IEEE Transactions on Industrial Informatics*, *17*(2), 732–745. Retrieved from https://doi.org/10.1109/tii.2020.3004343 doi: 10.1109/tii.2020.3004343

Petrlik, M., Vonasek, V., & Saska, M. (2019, October). Coverage optimization in the cooperative surveillance task using multiple micro aerial vehicles. In *2019 IEEE international conference on systems, man and cybernetics (SMC)*. IEEE. Retrieved from https://doi.org/10.1109/smc.2019.8914330 doi: 10.1109/smc.2019.8914330

PX4. (2022). *PX4 user guide.* From PX4 User Guide. Retrieved from https://docs.px4.io/main/en/

QGC. (2022). *QGC website.* From QGC - QGroundControl - Drone Control. Retrieved from http://qgroundcontrol.com

Rahmaniar, W., & Santoso, A. W. (2022). Sensor integration for real-time data acquisition in aerial surveillance. *Australian Journal of Electrical and Electronics Engineering*, *19*(2), 117-128. Retrieved from https://doi.org/10.1080/1448837X.2021.2023070 doi: 10.1080/1448837X.2021.2023070

Raptis, I. A., & Valavanis, K. P. (2011). *Linear and nonlinear control of small-scale unmanned helicopters*.

Springer Netherlands. Retrieved from https://doi.org/10.1007/978-94-007-0023-9 doi: 10.1007/978-94-007-0023-9

Ratnoo, A., Sujit, P., & Kothari, M. (2011, January). Adaptive optimal path following for high wind flights. *IFAC Proceedings Volumes*, *44*(1), 12985–12990. Retrieved from https://doi.org/10.3182/20110828-6-it-1002.03720 doi: 10.3182/20110828-6-it-1002.03720

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). *You only look once: Unified, real-time object detection.* arXiv. Retrieved from https://arxiv.org/abs/1506.02640 doi: 10.48550/ARXIV.1506.02640

Ren, W. (2006). Consensus based formation control strategies for multi-vehicle systems. In *2006 american control conference.* IEEE. Retrieved from https://doi.org/10.1109/acc.2006.1657384 doi: 10.1109/acc.2006.1657384

Ridao, P., Yuh, J., Batlle, J., & Sugihara, K. (2000). On AUV control architecture. In *Proceedings. 2000 IEEE/RSJ international conference on intelligent robots and systems (IROS 2000) (cat. no.00ch37113).* IEEE. Retrieved from https://doi.org/10.1109/iros.2000.893126 doi: 10.1109/iros.2000.893126

ROS. (2022). *ROS website.* From ROS. Retrieved from https://www.ros.org/

Rosalie, M., Dentler, J. E., Danoy, G., Bouvry, P., Kannan, S., Olivares-Mendez, M. A., & Voos, H. (2017, June). Area exploration with a swarm of UAVs combining deterministic chaotic ant colony mobility with position MPC. In *2017 international conference on unmanned aircraft systems (ICUAS).* IEEE. Retrieved from https://doi.org/10.1109/icuas.2017.7991418 doi: 10.1109/icuas.2017.7991418

Santos, L. (2021). *Controlo de voo de formação para missões de busca em ambiente marítimo com UAVs* (Unpublished master's thesis). Academia da Força Aérea Portuguesa.

Shojaei, K. (2015, September). Leader–follower formation control of underactuated autonomous marine surface vehicles with limited torque. *Ocean Engineering*, *105*, 196–205. Retrieved from https://doi.org/10.1016/j.oceaneng.2015.06.026 doi: 10.1016/j.oceaneng.2015.06.026

Shojaei, K. (2016). Observer-based neural adaptive formation control of autonomous surface vessels with limited torque. *Robotics and Autonomous Systems*, *78*, 83-96. Retrieved from https://www.sciencedirect.com/science/article/pii/S0921889016000129 doi: https://doi.org/10.1016/j.robot.2016.01.005

Skjetne, R., Moi, S., & Fossen, T. (2002, December). Nonlinear formation control of marine craft. In *Proceedings of the 41st IEEE conference on decision and control, 2002.* (p. 1699—1704). IEEE. Retrieved from https://doi.org/10.1109/cdc.2002.1184765 doi: 10.1109/cdc.2002.1184765

Soetanto, D., Lapierre, L., & Pascoal, A. (2003). Adaptive, non-singular path-following control of dynamic wheeled robots. In *42nd IEEE international conference on decision and control (IEEE cat. no.03ch37475).* IEEE. Retrieved from https://doi.org/10.1109/cdc.2003.1272868 doi: 10.1109/cdc.2003.1272868

Souza, R. M. J. A., Lima, G. V., Morais, A. S., Oliveira-Lopes, L. C., Ramos, D. C., & Tofoli, F. L. (2022, February). Modified artificial potential field for the path planning of aircraft swarms in three-dimensional environments. *Sensors*, *22*(4), 1558. Retrieved from https://doi.org/10.3390/s22041558 doi: 10.3390/s22041558

Suda, T. (2019). Construction of lyapunov functions using helmholtz–hodge decomposition. *Discrete &*

Continuous Dynamical Systems - A, *39*(5), 2437–2454. Retrieved from https://doi.org/10.3934/dcds.2019103 doi: 10.3934/dcds.2019103

Sujit, P., Saripalli, S., & Sousa, J. B. (2014, February). Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems*, *34*(1), 42–59. Retrieved from https://doi.org/10.1109/mcs.2013.2287568 doi: 10.1109/mcs.2013.2287568

Tao, G. (2014, November). Multivariable adaptive control: A survey. *Automatica*, *50*(11), 2737–2764. Retrieved from https://doi.org/10.1016/j.automatica.2014.10.015 doi: 10.1016/j.automatica.2014.10.015

Tazibt, C. Y., Achir, N., Muhlethaler, P., & Djamah, T. (2018). UAV-based data gathering using an artificial potential fields approach. In *2018 ieee 88th vehicular technology conference (vtc-fall)* (p. 1-5). doi: 10.1109/VTCFall.2018.8691007

Viguria, A., Maza, I., & Ollero, A. (2010, January). Distributed service-based cooperation in aerial/ground robot teams applied to fire detection and extinguishing missions. *Advanced Robotics*, *24*(1-2), 1–23. Retrieved from https://doi.org/10.1163/016918609x12585524300339 doi: 10.1163/016918609x12585524300339

Wang, B., Dong, X., & Chen, B. M. (2010, June). Cascaded control of 3d path following for an unmanned helicopter. In *2010 IEEE conference on cybernetics and intelligent systems.* IEEE. Retrieved from https://doi.org/10.1109/iccis.2010.5518579 doi: 10.1109/iccis.2010.5518579

Wang, P., & Hadaegh, F. (1996, 07). Coordination and control of multiple microspacecraft moving in formation. *Journal of the Astronautical Sciences*, *44*.

Wang, Y., Wang, D., & Zhu, S. (2019a). Cooperative moving path following for multiple fixed-wing unmanned aerial vehicles with speed constraints. *Automatica*, *100*, 82-89. Retrieved from https://www.sciencedirect.com/science/article/pii/S0005109818305363 doi: https://doi.org/10.1016/j.automatica.2018.11.004

Wang, Y., Wang, D., & Zhu, S. (2019b, February). Cooperative moving path following for multiple fixed-wing unmanned aerial vehicles with speed constraints. *Automatica*, *100*, 82–89. Retrieved from https://doi.org/10.1016/j.automatica.2018.11.004 doi: 10.1016/j.automatica.2018.11.004

Yu, X., & Liu, L. (2016, June). Distributed circular formation control of ring-networked nonholonomic vehicles. *Automatica*, *68*, 92–99. Retrieved from https://doi.org/10.1016/j.automatica.2016.01.056 doi: 10.1016/j.automatica.2016.01.056

Yun, B., Chen, B. M., Lum, K. Y., & Lee, T. H. (2010, January). Design and implementation of a leader-follower cooperative control system for unmanned helicopters. *Journal of Control Theory and Applications*, *8*(1), 61–68. Retrieved from https://doi.org/10.1007/s11768-010-9188-6 doi: 10.1007/s11768-010-9188-6

Zhang, M., & Liu, H. H. T. (2015, jun). Cooperative tracking a moving target using multiple fixed-wing UAVs. *Journal of Intelligent & Robotic Systems*, *81*(3-4), 505–529. Retrieved from https://doi.org/10.1007/s10846-015-0236-9 doi: 10.1007/s10846-015-0236-9

Zhou, W., Liu, Z., Li, J., Xu, X., & Shen, L. (2021, November). Multi-target tracking for unmanned aerial vehicle swarms using deep reinforcement learning. *Neurocomputing*, *466*, 285–297. Retrieved from https://

doi.org/10.1016/j.neucom.2021.09.044 doi: 10.1016/j.neucom.2021.09.044