Vector field formation control

Fixed-wing formation control based on time-varying vector field

Master of Science Thesis

Zobeer A. Mohammad

Main Supervisor: Simone Baldi

Vector field formation control

Fixed-wing formation control based on time-varying vector field

Master of Science Thesis

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 9Th December, 2020 at 10:00 AM.

Author:

Zobeer A. Mohammad

Student number:	4899024	
Project duration:	December 1, 2019 – November 15, 2020	
Thesis committee:	Dr. Sergio Grammatico,	TU Delft, Committee chair
	Dr. Simone Baldi,	TU Delft, supervisor
	Dr. ir. Arjan van Genderen,	TU Delft, Committee member
	Mr. Ximan Wang,	TU Delft, Co-supervisor

A special thanks to TuDelft for the great environment and Tesi Tecnologia Sicurezza that made possible this experience:

Keywords: Vector field, fixed-wing, formation control, formation flight.

Front picture: The cover picture captures a formation flight performed by the author (Z.A. Mohammad) during a military training session with Italian Air force.

It is much better to know something about everything
than to know everything about one thing.

Blaise Pascal

Contents

Abbreviations & Acronyms

VFF:	Vector field for formation control.
UAV:	Unmanned aerial vehicle.
VTOL:	Vertical Take-Off and Landing vehicle.
MEMS:	Micro Electro-Mechanical Systems.
ROS:	Robot operating system.
TCP/IP:	Transmission Control Protocol/Internet Protocol.
UDP	User Datagram Protocol.
ESC:	Electronic speed controller.
MPC:	Model Predictive Control.
PID:	Proportional Integral Derivative Controller.
LLC:	Low Level Controller.
PWM:	Pulse width modulation.
LAN:	Local area network.
RAM:	Random Access Memory.
RF:	Radio frequency.
SITL:	Software in the loop.
HITL:	Hardware in the loop.
GPS:	Global Positioning System.
UT:	Unicycle-Type approach.

Summary

The mission control of a fixed-wings UAV is slightly more complex compared to other models like quad-rotors or unicycles since the dynamic of a fixed-wing aircraft does not allow rapid direction changes or stop-rotate-go type of movements.
Those restriction needs to be taken into consideration when building a mission control system, moreover, the problem increases in complexity when the goal is to fly in formation with several UAVs, indeed during its development, we have to handle also the scalability of the communication link and the robustness of the formation during harsh weather conditions and disturbances.
In this research we propose a solution based on a vector field This vector field is computed based on the leader attitude and the followers positions.
From the vector field, the follower can calculate the heading and the speed that it has to keep in order to reach the formation.
To reach this solution, we first defined a desired heading and ground speed, then, by defining the goals of the UAV and the dynamic of the UAV we found the commanded heading and the commanded ground speed for each follower.
The stability of the result is checked by dividing the convergence problem into two smaller problems and applying the Lyapunov stability theory.
The solution is fully implemented using state of the art technology in embedded and autopilots for UAV. The system was physically simulated with ROS in Gazebo.
At the end, the performances are compared to another state of the art solution, in which VFF outperformed in most of the cases.

Structure of the thesis

The first part of the thesis focus on the background that is required to understand the problem and to follower the procedures to reach the solution. The first chapter 1 gives an introduction about the UAVs and focus on its challenges and applications, then it introduces some of the problems currently under research to understand their implication and importance in the current applications.
A detailed background is given in the chapter 2 that prepares the reader for the argument discussed in the following chapters. This section will gives all the mathematical knowledge and tools to be familiar with autonomous fixed-wing aircraft. Some more mathematical concepts are given to describe the problems and model the dynamics of UAV. Then, the current research result are discussed briefly, and at the end of the section, a light introduction is given about the frameworks used to implement our solution.
The main contribution of the thesis is in the following chapters, starting from 3 that introduces the vector field for straight and circular path following, then the formation control problem is introduced. Here we find a solution and we analyse its stability using mathematical frameworks.
The control laws obtained from this chapter are implemented in chapter 4, which contains most of the work related to the embedded side of the project: the first step is the explanation of the new software architecture for the autopilot that is designed for the followers based on the requirements given previously. Once the system is fully implemented, it was evaluated in the section 5 that contains all the simulation and comparison with other control laws.

1

Introduction

A short introduction about the UAVs is given in this chapter, with a focus on challenges and applications, then we will introduce some of the problems that are still under research and understand their implication and importance in the current applications. At the end we will do a short introduction to the proposed solution for formation control based on vector field, then how we tested our solution and the frameworks that were adopted to solve the problems.

Unmanned Aerial Vehicles (UAV) or commonly knows as drones, are aerial vehicles without a human pilot on-board. The flight of a UAV can be operated using a remote control station by a human or completely autonomously by an on-board computer.
Inside the UAVs category, there are many different type of vehicles, optimized for specific application, the three main category are Fixed-Wing, multi-copter and Vertical Take-Off Landing (VTOL).
A multi-rotor is an aircraft with more than two lift-generating rotors, that produce lift just by the propellers spinning. The main advantages of this configuration are an ease construction, since unlike helicopters, the do not have a variable pitch propellers for flight stability. The manoeuvrability is achieved by varying the rotation speed of each rotor.
An example of multi-rotor is the most knows quadcopter: it has 4 propellers at the edge of a cross. The quadcopter control the motion by changing the speeds of its propellers as visible in figure 1.1.

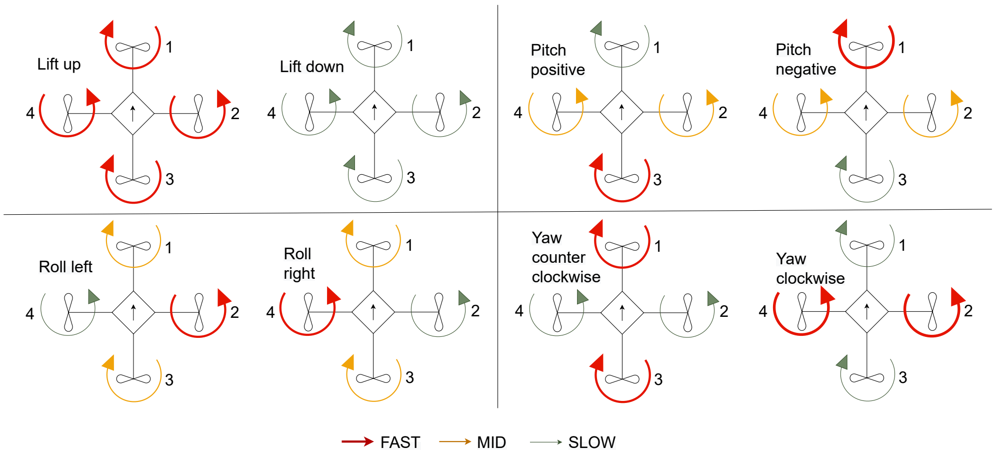

Figure 1.1: Quadcopter motion: lift up is made possible by increasing the rotation speed of every propeller, the lift down is similar but decrease the rotation speed. Pitch positive can be performed by increasing the speed of propeller 3 and decreasing the speed of propeller 1, inversely, the quadcopter will do a pitch negative. The rolling is achieved similarly to the pitch but working on propellers 2 and 4. The counter clockwise yaw is achieved by increasing the counter clockwise propellers speed and decreasing the other two.

Several formation formation paper are based on this assumption that the agent can move freely in the space in any direction, this can work with the control of a multi-rotor, but the same assumption is not valid in case of fixed-wing since they can take off vertically and keep a fixed position in the space.

The fixed-wing UAV is an aircraft that is capable of flight using the lift force caused by the aircraft's forward airspeed and the wings. The wings are shaped such that they can produce a differential pressure, pushing the airplane upwards; some panels above the wings or on the tail are used to deflect the air and control the attitude of the plane.
The main advantage of this configuration is that its possible to fly by using a very less amount of energy or totally energy free by using the air flow.

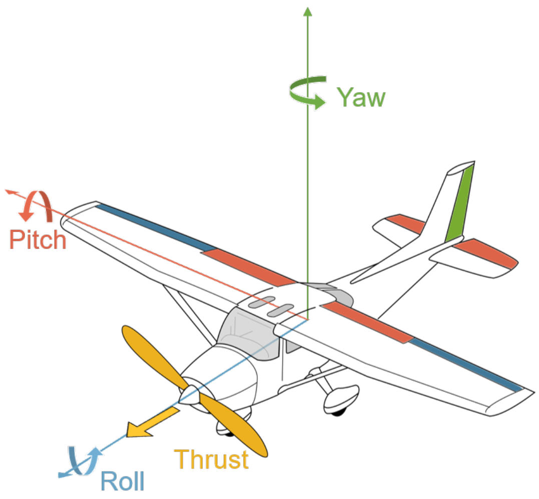

Figure 1.2: A fixed-wing plane changes its attitude using some moving panels on his wings or on the tail.

These characteristics makes the fixed-wind UAV a key technology for several application like mapping, patrolling and search and rescue [1].
Moreover a group of UAVs can be used to cover a wider area since a close formation flight lead to a potential fuel saving [2].
The dynamic of a fixed-wing plane is quite different from the one previously described, without going into mathematical details, a fixed-wing UAV need a forward motion speed to keep the vertical lift, this means that lateral motions are not feasible and also a backwards motion will make the plane crash.
On a common fixed-wing plane, the propellers are used just to create thrust differently from multi-rotor plane which can use the propellers also for turning or lifting. The attitude of the plane is instead controlled by some mobile panels on wings and a rudder on the tail of the aircraft.

1.0.1. Challenges and application of UAVs

In recent years, their use is rapidly growing in recreational, scientific, military, commercial, agricultural and many more applications. For instance, UAV are used for aerial photography: to analyse buildings, surveillance [3] or analyse solar panels malfunctions [4]. There are several research going on to extend the usefulness of UAVs beyond their current capability, increase the flight range and the autonomy of some operation are one of the current main focus. The huge progress in electronics miniaturization and control field allows to reduce the size of UAVs such that the mass and the volume is decreased, this significantly impact its energy consumption.

A step to increase the flight time of UAVs was taken by the introduction of fuel cell propulsion that has a high energy density [5]. On the other side, the fuel cell technology require an higher amount of size and volume to produce enough power for the UAVs thrust, thus an bigger size of the plane and more drag. further research is going on to reduce the size of this propulsion.

The characteristic bird fly have inspired many solution to increase the efficiency of UAV, one of this is the formation fly, since it is used by birds during long migratory trips.
In a formation fly, the air deflected from the leader helps the followers to support their own weight and decreases the drag force. Moreover this flying formation in nature can increase theoretically the range up to 71% [6].
Flying in formation requires a high level of training, even for professional pilots is a complex task that requires high level of concentration for a long period since the followers are subject to air turbulence generated from other planes. Thus is important to develop an automation system that can carry our this task autonomously by correcting the trajectory many more times than a pilot can do.
Moreover, since a intercontinental flight can take more than 10h for 20000km, the system has also to be independent from a ground control station and be based just on the data shared between the agents.
Recently in the civil aviation, Airbus demonstrated that an aircraft can save up-to 10% of fuel during a formation flight this will reduce the aviation's environmental footprint, an approach that can be carried out without additional equipage either on the aircraft or on the ground [7]. Several tests of intercontinental flights in formation are currently going on.

Research question: *How we can autonomously control UAVs flying in formation?*

The formation control of mobile robots and fixed-wing UAV have some similarities like collision avoidance and communication constraints, however the UAVs has a different dynamic and it needs to have a forward motion in order to generate the required lift to fly, thus stop-and-wait, stop-and-reverse and instantaneous lateral motions are not possible.

In this thesis we will propose a new formation algorithm based on the leader-follower approach, which can keep theoretically unlimited amount of fixed-wing UAV in formation. The solution is studied for UAVs dynamics and it allows the followers to keep the formation during any leader's movement and any physical feasible trajectory.
The formation can assume any shape, until the communication link is stable, and it is defined by a list of offset vectors $P = \{\vec{P}_1, \vec{P}_2 ... \vec{P}_n\}$ where n is the number of followers and the vectors lay in the leader frame.
The proposed solution rely on local information shared by the leader, like its position, ground speed and heading, plus some mission-details provided at the beginning by a ground control station, thus once the mission in started, the followers are totally independent from each other and they rely just on a communication link with the leader, that can be connected with the ground control station to obtain updates about the mission plan.
By defining some light restrictions for the maximum speed and turning rate of the leader, that they have to be lower than the follower's, it is possible to provide a proof of convergence based on Lyapunov functions.

At the end of the thesis, the solution is fully implemented in a virtual environment using Gazebo as physical simulator, PX4 as low level control for the plane attitude and ROS as framework for the communication.

2

Background

This chapter aims to prepare the reader for the arguments discussed in the following chapters. To understand those, the reader must be familiar with a wide range of topics.
To describe the dynamics of a UAV, some concepts of coordinate frames and forces are described in the section 2.1. Thus, after these concepts, the modeling of a fixed-wing UAV is introduced in section 2.2.
The section 2.3 introduces the communication limits and the framework used in our solution. Then, the section 2.4 will introduce the core of this thesis, and the previous work that is done for the formation of autonomous agents, we will analyze their downsides and try to understand their limits if applied to fixed-wing UAVs.

2.1. Coordinate Frames

In this section we will see how to do the description of the UAV bodies orientation relative to each other elements such an example, we have to describe the UAV position and orientation relative to the earth, we may also need to know how sensors are oriented relative to the UAV body and how the other UAV are oriented relative to each other.
A different coordinate frame simplifies the description of some motions and forces that can not be described easily in a fixed inertial reference frame. On the other side, the mission requirements (way-points and flight trajectory) are specified in the inertial frame.

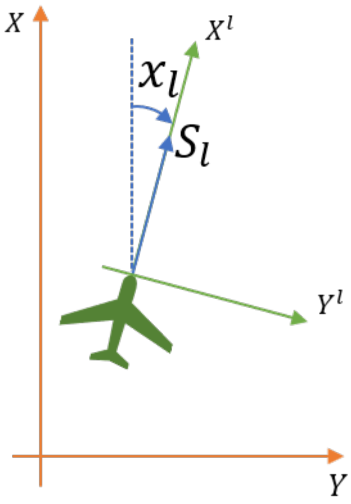

Figure 2.1: Inertial frame of the UAV in orange, with a leader in green. Also the leader frame is shown in green. The frame generated x^l and y^l is the leader frame.

The green frame showed in figure 2.1 is called body plane and in this case, it is referred to the leader. This frame is quite important for our proposed solution since it simplifies many calculations

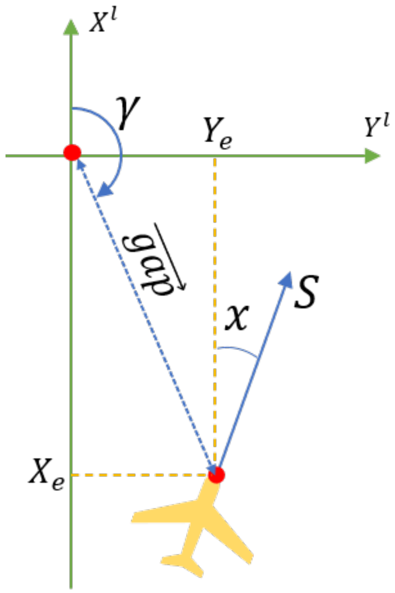

Figure 2.2: The follower UAV (yellow) in the leader frame (green), the red point is the position of the leader, the vector $\overline{gap}$ from the leader position indicates the point that the follower has to follow to keep the formation.

- χ_l is the angle between the axis X in the inertial frame and the axis X^l of the leader frame. Physically, these angles indicate the direction of the fly of the leader in the inertial frame.
- χ is the same angle of χ_l but for the follower, and it indicates its heading.
- γ is the angle between the axis X in the inertial frame and the vector that links the leader position and the follower position in the inertial frame.
- S and S_l are ground speed vectors.
- The distance between the follower and the leader is d.

2.1.1. Airspeed, Wind Speed, Ground Speed

In order to control the UAV properly and keep the formation, some information about the UAV and the environment is needed.
Since the UAV is in the air, the estimation of its speed is not an easy task without any fixed reference. Wind speed is the speed of the air mass in which the UAV is flying, the ground speed is the speed of the UAV relative to a fixed ground position (inertial frame). Airspeed is the plane speed relative to the air, it is commonly measured using a Pitot Tube. Since the fixed-wing UAV generate its lift from the air over their wings [8], if the airspeed is too low, the plane will stall. Moreover, in order to keep the correct ground track, the autopilot system has to correct the UAV heading based on wind speed, wind direction, and airspeed.

2.2. Fixed-Wing UAV Model

In the development of this thesis, we have to keep in mind the system architecture shown in Figure 2.3.

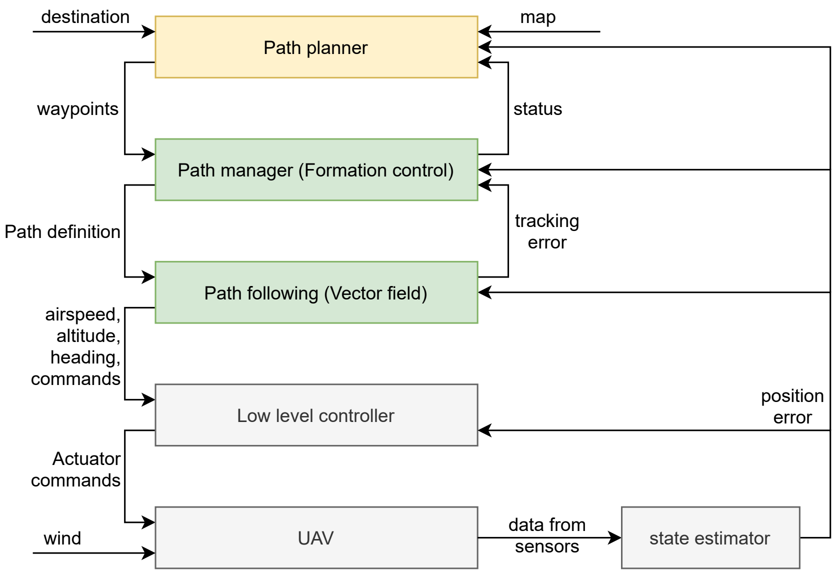

Figure 2.3: A simplified UAV control architecture and the data exchanged between the layers, we focus our solution in the green layers. For the gray layers that implements all the low-level attitude control, a generic control software (PX4) for UAV is used. The path planner layer is implemented using a standard ground control station. This control scheme is based on the one proposed in [9]

The system is divided into layers, this creates an abstraction between the different domains needed to control a UAV, all the blocks are interconnected by interfaces. The first layer is the Path Planner, this layer receives in input the mission, composed of one or more destinations and/or a map. This layer then calculates the points on the map that shall be reached and crossed by the UAV, called way-points.
The way-points are then sent to the Path Manager, this layer contains several algorithms that compute the route to reach the way-points. This path is then sent to the path following algorithm that calculates the commanded attitude for the UAV, these are sent to the low-level controllers that will compute the actuators commands, for instance, move a flap, or increase the speed of the motors.
The state estimation is the part of the control stack that provides an estimation of the UAV states to each layer such that they can estimate the error. A typical UAV is equipped with several sensors that are used by the control loops, those sensors are usually:

- Accelerometers: they are typically integrated into a MEMS sensor along 3 axes and they return a value proportional to the acceleration. These sensors are widely used to calculate the attitude of the fixed-wing UAV since they can estimate the earth's gravitational force vector and its decomposition along the x and y-axis. Unfortunately, due to the vibration introduces by the air turbulence and motor vibration, the measure of the acceleration is subject to a noise that can be approximated to Gaussian noise.

- Gyroscope: It is a MEMS sensor that works based on the principle of the Coriolis acceleration. It returns the angular rate of the three rotational axes, also this sensor is subject to a noise that is less than the accelerometers. The angular rate can be integrated to obtain the angle, but this will lead to a drift in the time since also the small noise will be integrated. A better solution is to mix the data coming from the accelerometer and the gyroscope using a filtering and sensor fusion algorithm like the Kalman filter.
- Pressure sensor: It is a sensor that returns a value proportional to the air pressure. Based on its configuration, it can be used as an indication of the altitude if it measures the absolute pressure, or if it is differential pressure it can be used to estimate the airspeed of the aircraft.
- Compass: this sensor can measure the earth's magnetic field, but due to the interference generated by the electric motors, the value returned by the sensor is highly noisy. The measurement can be improved a lot by using a Kalman filter and combining the measurement with a gyroscope that measures the yaw movement of the UAV.
- Global Positioning System: is a satellite-based navigation system that provides 3D positioning information close to the earth's surface. Due to the synchronization errors between the sensors and the satellites, physical variation of the ambient and satellite orbit errors, at least 4 satellites are required to get a precision around 4mt. ([10])

A state estimator receives the data from several sensors and estimates the position and attitude of the UAV, this data is then sent to all the previous layer to estimate the position error and perform error correction.
For the control and simulation purposes, we will use a simulated UAV that needs to catch all the main dynamics of the physical UAV. The sensor will be simulated too and noise is introduced to each of them to simulate a realistic application. A simulation UAV model is usually nonlinear and is mathematically complex. Therefore, to simplify the design and the proof of the controller, a simplified dynamic is used, then it is validated on a more realistic physical simulator. Moreover, several dynamic models can be used for different control goals: for the low-level control of the UAV attitude, a detailed model of the UAV dynamic can be used, but for the high-level guidance control system, like path planning or formation control, a simplified dynamic of the vehicle can be adopted.
Since the focus of this thesis is the high-level control, the dynamics for the low-level controller are not introduced, but they can be found widely in the literature [11] [12]. Also, the low-level controllers are not part of this thesis, we will suppose that the UAV is equipped with a controller for attitude and hold loops for speed and altitude dynamics and that the low-level controllers are proper and the UAV is fully controllable.
Our work will mainly be on the Path following and Path manager layers which controls mainly the direction and position of the UAV on a high-level view, thus we can adopt a simplified dynamics that we will call High-level dynamics.

The high-level dynamic of the UAV can be expressed as:

$$\begin{aligned} \dot{x} &= V\cos(\psi) + W_x \\ \dot{y} &= V\sin(\psi) + W_y \end{aligned} \tag{2.1}$$

(W_x, W_y) represent the wind components in x and y axes respectively, V is the airspeed and ψ is the heading.

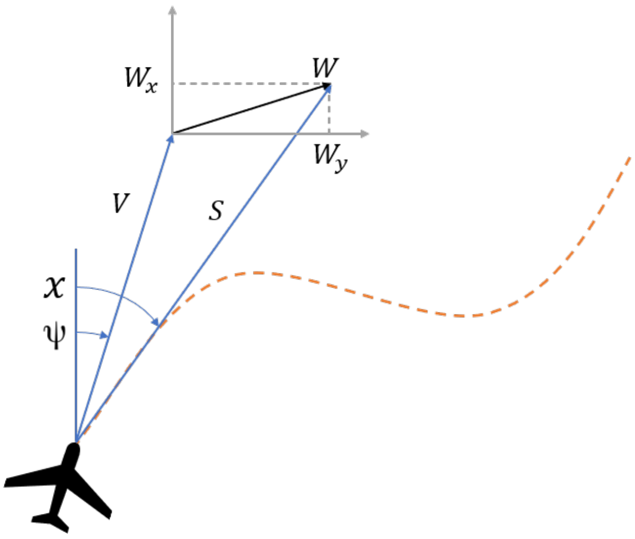

Figure 2.4: A UAV flying in a windy space, the ground track is the orange line. The relationship between the airspeed V_a, the windspeed W and ground speed V_g, as well as the relationship between heading γ and course χ

The equation (2.1) can be also be expressed in terms of the ground speed S:

$$\begin{aligned} \dot{x} &= V_x + W_x = S_x = S\cos(\chi) \\ \dot{y} &= V_y + W_y = S_y = S\sin(\chi) \end{aligned} \tag{2.2}$$

in which χ is defined as the ground heading.
We can then define the equation motion for χ and S:

$$\dot{\chi} = \alpha(\chi^c - \chi) \tag{2.3}$$

where α is a parameter that is related to the yaw rate of the UAV, and χ^c is the commanded course that depends on the control algorithm.

2.3. Communication between UAVs

The UAVs in formation fly needs to exchange information for trajectory control and collision avoidance. For instance, the leader of the formation will transmit its heading, speed, and flight altitude to the followers such that they can calculate the proper action to keep the

formation. Moreover, in the proposed solution, the leader transmits also the commanded heading value, which is used by the followers to anticipate the air maneuver of the leader and keep a stable formation also during turns.
A good architecture for the communication between the UAVs shall be scalable, reliable, and decentralized.
If we increase the number of aircraft flying in formation, the number of data transferred in this channel is and the links between the UVAs will increase, the number of links mustn't increase exponentially. In our proposed solution, we are not considering collisions between the followers, thus the only communication link is between the leader towards each follower, thus for a new follower, there is just one new connection.
The transmitted information shall be reliable since an error during the transmission could change a value and cause an accident, for instance, if a follower receives a wrong heading, it will break the formation and cause a dangerous situation for the other followers and for the leader. To overcome this problem, in the proposed solution, we will use the ROS framework for communication, which uses TCP/IP. This means that the received data is verified and in the case, it is corrupted, a correction algorithm will try to recover it, but in the case, too many bits are corrupted, the data is discarded and the leader will transmit again the missing packet.
A communication system between UAVs shall be also fault-tolerant, this can be achieved in different ways, by using a redundant communication system or by making it distributed, such that if a follower has a fault during the transmission, it does not affect the other follower's reception.

2.3.1. ROS and PX4 framework

The proposed solution is based on the ROS Communication framework that is widely used in embedded, robotics and industrial applications to share data between several agents called nodes.
ROS is based on the publisher/subscriber approach, which are the two main roles during a communication.
The data that needs to be transmitted is called messages and their definition is specified before the execution of the nodes. A message can be composed of several types of formats like integers, arrays, float values, strings, or other messages.
All the nodes and the topics shall be registered to a core that handles all the communication and visibility of the whole communication system.

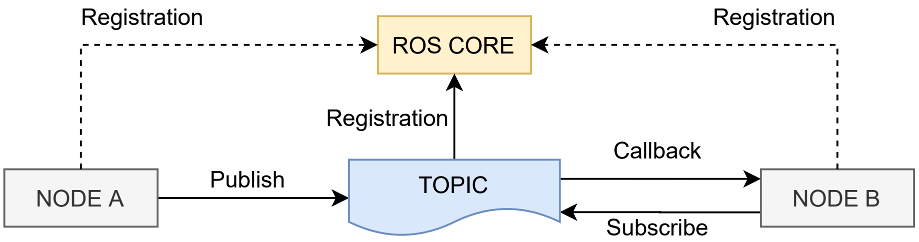

Figure 2.5: Representation of how ROS works, the gray boxes are the nodes that can publich or subscribe to a topic (in blue). All the parts in this system shall be registered to the ROSCORE in order to communicate to each other.

For each message, one or more topics are defined. A topic can be considered as a whiteboard where each agent can post a message or read a message. An unlimited number of topics can be created and for each new message published on a topic, the previous content is overwritten.
The PX4 framework is an open-source autopilot system used in a wide range of cases, from drones to industrial applications. It is also used in the research platform for drones since it can be easily integrated into a third part system as we did in our project. The PX4 architecture might look complex at first sight, but it's made of two main components: middleware blocks on the top and the flight stack on the bottom. The middleware blocks offer complementary functions to the flight system, such as the Storage block that provides data storage, parameters storage, logging data, and flashcard storage. The two blocks external connectivity and drivers provide the flight control stack with a wide communication capability. In our system, we will use MAVLink as a protocol to share data between the PX4 flight stack, the companion hardware, and the simulation.
All the blocks in the software architecture of the PX4 are interconnected with a virtual message communication called uORB that provides a reliable data share between all the components of a system. The flight stack contains all the stacks described in the section 2.3. In our application, we are going to use the attitude and rate controllers, the output driver, the position and attitude estimator and the sensor hub. While in simulation, the Driver block will get the data from the simulating environment instead of real sensors.

2.4. Formation Fly

The problem of formation fly is a mix of several tasks, one is to start from a point and reach in the most efficient way another point. The second is to keep as much as possible the shape during the fly, the third is to avoid obstacles and the fourth is to make the system scalable such that the number of links does not increase exponentially or even linearly with the number of UAVs in the formation. The fifth is to make the system robust to faults, this means that the computation shall be distributed between the agents and that they should not rely on a single centered computation unit.
In this thesis, we will try to take into consideration all those points except the obstacle avoidance, and find a solution that can satisfy most of them. We can split our main research

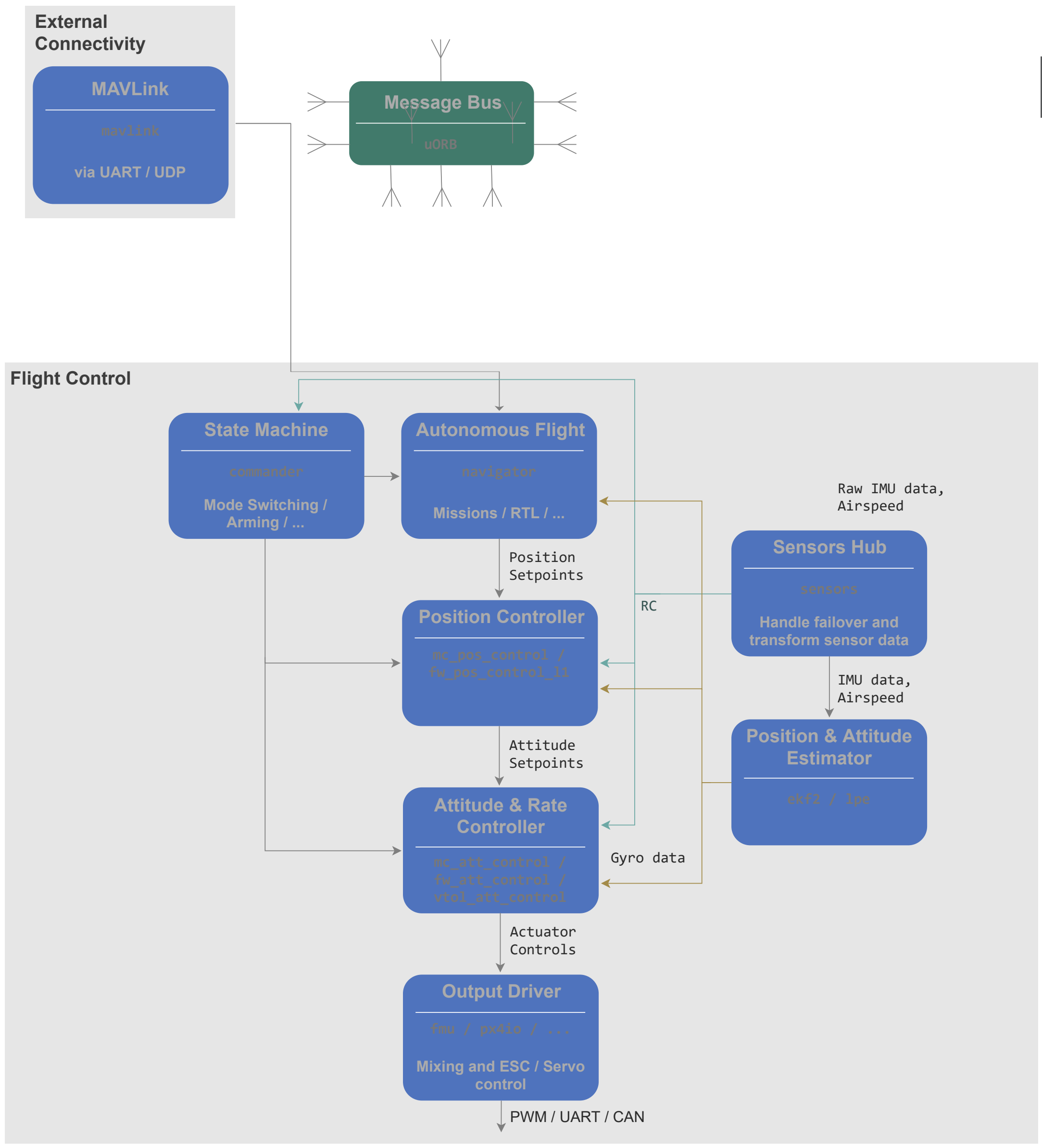

Figure 2.6: High-level software architecture. Source: PX4 developer guide

question 1.0.1 in two sub-questions regarding the task introduced before, first:

Research sub-question: *How we provide appropriate velocity and heading dynamics for formation control?*

Where appropriate means that the UAVs shall converge into a predefined formation shape in a limited amount of time and that they have to keep this shape. And from a more embedded prospective:

Research sub-question: *How we can provide a scalable and robust architecture for formation control?*

The scalability requirement makes necessary the adoption of a decentralized system, this is also what happens in nature, since a bird is not expected to watch all the other bird in the group.
In literature, the formation control problem of multiple mobile vehicles is widely discussed in the robotics and control community.
Four main categories of structure approaches that can be named are [13]:

- Behavioral approach: it consists of prescribing several desired behaviors for each agent and to calculate the control action individually based on a weighted average of the control that satisfies each behavior [14]. This is a decentralized control method, thus there is less communication involved and it is more robust against faults that can happen to a centralized system and its communication. The main disadvantage of this architecture is that the mathematical analysis of this approach is complex. [15]
- Virtual Structure Approach: the entire formation is considered as a single structure. Every follower is controlled by a central device that command the whole fleet as a single body [16]-[17]. The whole system is based on a central device that could fail, thus make the system not applicable is mission-critical applications, moreover, the main advantage is that a single mathematical rule translates the sensory input space into actuators input without defining behaviours or rules. Moreover, robustness and stability can be proven using theories from physics, control theory, and graph theory. [18].
- Leader-follower: One of the agent is designated as leader that get a designated path and the rest are followers and they have to maintain the shape of the formation by tracking the position and orientation of the leader with some offset. One of the main disadvantage of this architecture is that it require a centralized communication link between the leader and all the followers, which makes it less fault-tolerant, moreover there is not a feedback communication from the followers to the leader. For example, if the leader is moving too fast and the follower can not catch up, it will breaks the formation [19]. On the other hand, the architecture is very simple to control and implement since the computation is distributed over each of the flying agent, making the system easy to scale, moreover some of the disadvantages can be fixed by few assumptions like lowering the maximum leader speed below the maximum follower speed.

Each of these methods has some disadvantage, in our solution, the architecture that is used is the leader-follower since it provides the best trade-off between simplicity, fault-tolerance, and scalability.
Many control schemes was proposed that can be used for formation control: PID based formation control was first introduced in [20], adaptive output feedback to a decentralized control approach was introduced in [21], a 3D potential field control method is discussed in [22].
But some of those methods were developed for dynamic vehicle without considering the constraints of a fixed-wing UAV: turn rate and minimum airspeed. Thus, some solution based on Model Predictive Control were proposed since they can handle with simplicity the optimization problem with constraints. MPC is a feedback control scheme in which the input to the system is calculated based on an optimization for each time step [23], the optimization of trajectory is based on a cost function that can take into account also the distance between the followers (thus find a trajectory that decreases the possibility of collisions) or distances between obstacles on the path.
Other studies focus on adaptive laws to estimate unmodelled dynamics of flying such as [24], in which adaptive laws are used to estimate wind knowledge and dynamics of the UAV model, or also the adaptive approach used in [25] where the idea is that each agent must converge to the model defined by its hierarchically superior neighbors, using a distributed inverse dynamics structure to compensate for the presence of uncertainties, estimation laws are devised for such matching gains, leading to adaptive synchronization.

3
Vector Field for Formation Control

In this chapter we are explaining the vector field and how it is used for path following. We will provide two vector field law for straight path following and circular paths. Then we will introduce the formation control problem and focus on the vector field formation control: in the first step we will introduce the problem, then provide some mathematical tool to describe the problem and reach a solution, we will provide a control law by defining the desired behavior of the followers and calculate the commanded ground speed and heading. The final step will be to give proof of its stability by splitting the problem into two subproblems.

The vector field is an assignment of a vector to each point in a subset of space. A vector field applied in a 2D subspace can be visualized as a collection of arrows with defined magnitude and direction. Vector fields can be used to model the heading of the UAV given some coordinates, the length of the arrow will represent proportionally the speed of the UAV. This shall not be confused with other methods [26] that uses the vector field to describe the gradient of a potential field to navigate through obstacles and approach their targets. In this thesis, the vector field is not considered to be static but it will be calculated continuously in real-time by the UAV based on several conditions that can be:

- The weather conditions such as wind or storms: the vector field can be calculated also by taking into account the effect of the wind or storms on the UAV. For each point, the UAV will calculate which heading command is the most appropriate to take to reach the goal or have proper ground tracking.

- The tracking object position and direction: the vector field can be generated to follow a line or an orbit path, by mixing them is also possible to create any shape of the path. Moreover, later we will use the vector field to follow a moving point.

3.1. Vector field for path following

A classic path for a UAV can be decomposed into two main objects: lines and orbits. Thus, just two control laws are required for the path follower system to track the ground path.
For instance, the path manager will get a list of waypoints that the UAV should touch, pass near-by or loiter around, then the path manager will decompose the path into lines and orbits and send them to the path follower which will calculate the vector field based on some laws and get the attitude, speed and heading command.

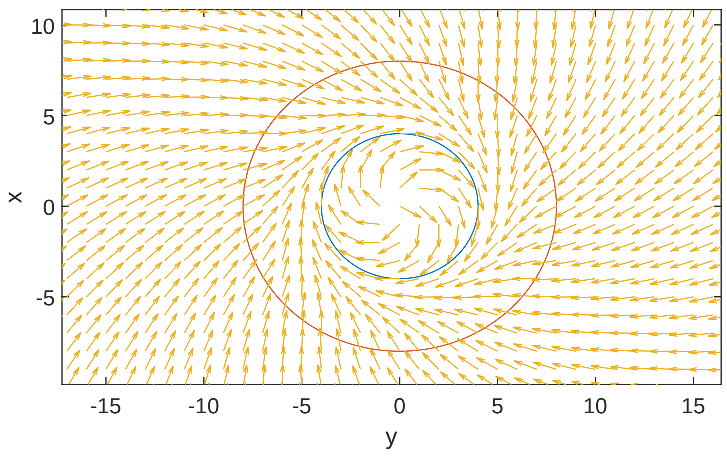

Figure 3.1: An example of a vector field generated to follow an orbit (blue line). The bigger orange circle is the boundary after which a different control law is applied.

In order to analyze the vector field approach, we will define two primitives path types: straight lines ad circular orbits. Those are easily extended to paths composed of multiple segments of arcs, orbits and lines.

3.1.1. Straight-line path following and orbit path

We consider two points A and B, which needs to be reached, thus we can define a straight path that crosses those points.
The control law that generated the waypoints will need to know the position, the speed, and the heading of the UAV to calculate the commanded heading.
Let L be the line between these two waypoints and ϵ be the lateral distance between L and the UAV, χ be the difference between the direction of the path and the course angle of the UAV. The goal of the control law is to set the heading of the UAV equal to χ^{∞} when y is high, and as it approaches zero, also the angle χ shall approach zero.
The paper [9] propose one of this possible solution:

$$\chi^d(y) = -\chi^{\infty}\frac{2}{\pi}\tan^{-1}(ky) \tag{3.1}$$

where k is defined as a positive constant that influence the transition rate of the commanded heading from χ^{∞} to zero. By considering $\chi^{\infty} \in (0, \frac{\pi}{2}]$, then:

$$-\frac{\pi}{2} < \chi^{\infty}\frac{2}{\pi}\tan^{-1}(ky) < \frac{\pi}{2} \tag{3.2}$$

for every y. we can use the Lyapunov function $W_1(y) = \frac{1}{2}y^2$ to prove that if $\chi = \chi^d(y)$ then $y \to 0$ asymptotically:

$$\begin{aligned} \dot{W}_1 &= V_g y \sin(\chi^d(y)) \\ &= -V_g y \sin(\chi^{\infty}\frac{2}{\pi}\tan^{-1}(ky)) \end{aligned} \tag{3.3}$$

which is negative for $y \neq 0$. Now we define $\tilde{\chi} = \chi - \chi^d(y)$ and its differentiation is then:

$$\begin{aligned} \dot{\tilde{\chi}} &= \dot{\chi} - \dot{\chi}^d(y) \\ &= \alpha(\chi^c - \chi) + \chi^{\infty}\frac{2}{\pi}\frac{k}{1+(ky)^2}V_g\sin(\chi) \end{aligned} \tag{3.4}$$

By defining another Lyapunov $W_2 = \frac{1}{2}\tilde{\chi}$ we obtain that:

$$\begin{aligned} \dot{W}_2 &= \tilde{\chi}\dot{\tilde{\chi}}(y) \\ &= \tilde{\chi}(\dot{\chi} - \dot{\chi}^d(y)) \\ &= \tilde{\chi}(\alpha(\chi^c - \chi) + \chi^{\infty}\frac{2}{\pi}\frac{2}{1+(ky)^2}V_g\sin\chi) \end{aligned} \tag{3.5}$$

To get this negative defined, we can define χ^c such that:

$$\chi^c = \chi - \frac{1}{\alpha}\chi^\infty \frac{2}{\pi}\frac{k}{1+(ky)^2}V_g \sin\chi - \frac{k}{\alpha}\mathrm{sat}(\frac{\tilde{\chi}}{\epsilon}) \tag{3.6}$$

where $\mathrm{sat}(x)$ is a saturation function defined to avoid scattering:

$$\mathrm{sat}(x) = \begin{cases} x & \text{if} \quad |x| \le 1 \\ sign(x) & \text{otherwise} \end{cases} \tag{3.7}$$

and $\epsilon > 0$ defines the boundary region.
Given this assumptions, $\dot{W}_2 \le -k|\tilde{\chi}|$.

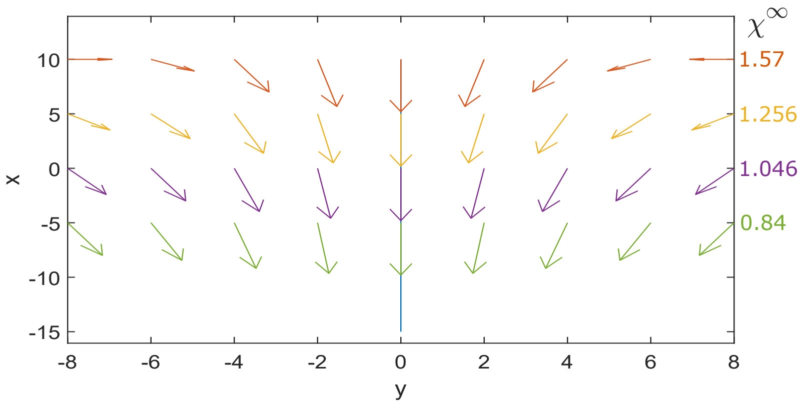

Figure 3.2: An example of a vector field generated in various positions of the UAV, with 4 different values of χ^∞. A higher value makes the system converge to the straight line more perpendicularly.

Similarly, as we did with the straight line, we can find a control law also in the case of a circular path. We assume that the center of the orbit (C_x, C_y) and the radius is known. When the distance of the UAV from the center of the orbit is high, it is expected for the UAV to fly towards the center, and when it gets closer, the heading of the UAV should equal the angle of a line tangent to the orbit.

$$\chi^d \approx \gamma - \pi \tag{3.8}$$

where γ is the angle of respect the center of the orbit and the UAV. When the distance is equal to the radius $d = r$, the UAV heading should be $\chi^d = \gamma - \frac{\pi}{2}$.

Thus, the vector field law can be generated by the following desired course:

$$\chi^d(d) = \gamma - \frac{\pi}{2} - tan^{-1}(k(d-r)) \tag{3.9}$$

Where k is a parameter that defines the change rate between $\gamma - \pi$ and $\gamma - \frac{\pi}{2}$. We can argue that $\tilde{d} = (d - r) \to 0$ asymptotically when $\chi = \chi^d(d)$ by using the Lyapunov function

$W_1 = \frac{1}{2}\tilde{d}^2$. By following the steps as we did previously in the straight line (also widely discussed in [9]), we obtain that $\tilde{\chi} \to 0$ if:

$$\chi^c = \chi + \frac{V_g}{\alpha d}\sin(\chi - \gamma) - \frac{\beta}{\alpha}V_g\cos(\chi - \gamma) - \frac{k}{\alpha d}\mathrm{sign}(\tilde{\chi}). \tag{3.10}$$

where β is defined as $\beta = \frac{kr}{1+((kr)\tilde{d})^2}$. and the sign function is substituted with the saturation function to avoid scattering.

3.2. Vector field formation control

A similar vector field approach can be used to solve the problem of the formation control. In this formation control problem, we will define a UAV Leader as:

$$\begin{aligned} \dot{x}_l &= S_l\cos(\chi_l) \\ \dot{y}_l &= S_l\sin(\chi_l) \\ \dot{\chi}_l &= \alpha_l(\chi^c - \chi_l) \end{aligned} \tag{3.11}$$

The leader's dynamic is similar to the one defined for in the equation (2.2), but the ground speed needs to be assumed smaller than every other follower $S_l < S$ such that they can reach the gap position in a finite amount of time. The leader will receive the mission plan from the ground control and define a path to follow. All the followers will track a point that is shifted by a vector $\overrightarrow{gap}$.

This new point will move into the 2D plane with speed and heading that is equal to the leader UAV. Thus, the new law can be based on the attitude data transmitted by the leader.

From a theoretical point of view, this problem can be seen as an orbit path following when the follower is far from the leader and a line path following problem when the follower approaches the set-point defined by the sum of the $\overrightarrow{gap}$ and the leader position.
But this simplification will not always give the expected behavior: first, if we consider the follower far from the leader, the equation (3.9) converges to:

$$\chi^d(d) = \gamma - \pi$$

Thus, as expected the follower will proceed straight to the direction of the leader plus the $\overrightarrow{gap}$.
Secondly, we consider the follower close to the goal position, then we can assume the problem as a straight-line path following problem, theoretically, it could work, but in a real application, the follower will not be able to keep his goal position in case of turns or path variation by the leader and it will always be lagging behind due to the physical delay introduced by the communication, by the mechanical response of the actuators and the flying physics.
To solve these issues, the follower should have a vector field law that anticipates the leader position. In our solution, this anticipation is made possible by taking into account the heading command from the leader to calculate the follower's heading command, such that, when the followers are close to their goal point, they perform the same operation done by the leader plus a correction.

3.2.1. Formulation of the solution

In equations (2.2) and (3.11) we defined the simplified dynamic models for the followers and the leader. Considering this situation, the objective of the control law is to calculate χ^c and S^c such that d reaches zero, the χ reaches the desired χ^d, and to maintain the setpoint, the follower should also reach the same speed of the leader this means that $S \Rightarrow S^l$.

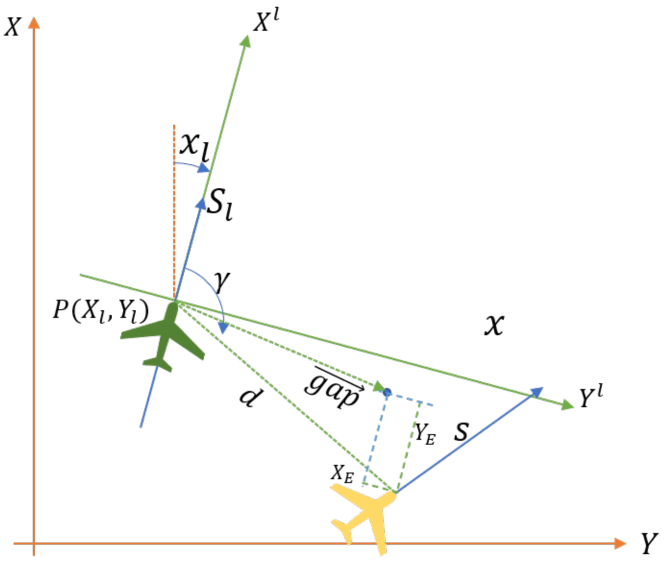

Figure 3.3: The leader's position is the inertial frame is $P(x_l, y_l)$, while the setpoint for the follower is a sum of the leader position plus a $\overrightarrow{\text{gap}}$ with an angle of γ in the leader frame, y_e and x_e is the follower error to reach the formation. d is the distance between leader and follower

We first need to express the follower position in polar coordinates in the internal frame, this will be useful for proof and to understand the dynamic of the model, the equation (2.2) can be expressed in terms of d and γ.

$$x = x_l + d\cos(\gamma) \tag{3.12}$$

$$y = y_l + d\sin(\gamma) \tag{3.13}$$

(x_l, y_l) are the coordinates of the leader, while (x, y) are the coordinates of the follower.

We can now obtain the dynamics for γ and d in these coordinates by deriving the previous two equations and substituting x and y with the dynamic of the follower defined in equation (2.2):

$$\begin{cases} S\cos(\chi) = \dot{x}_l + \dot{d}\cos(\gamma) - d\sin(\gamma)\dot{\gamma} \\ S\sin(\chi) = \dot{y}_l + \dot{d}\sin(\gamma) + d\cos(\gamma)\dot{\gamma} \end{cases} \tag{3.14}$$

To obtain the dynamic for $\dot{d}$, we will multiply the first equation to $\cos(\gamma)$ and the second equation to $\sin(\gamma)$, and we obtain:

$$\begin{cases} S\cos(\chi)\cos(\gamma) & = \dot{x}_l\cos(\gamma) + \dot{d}\cos(\gamma)\cos(\gamma) - d\sin(\gamma)\cos(\gamma)\dot{\gamma} \\ S\sin(\chi)\sin(\gamma) & = \dot{y}_l\sin(\gamma) + \dot{d}\sin(\gamma)\sin(\gamma) + d\cos(\gamma)\sin(\gamma)\dot{\gamma} \end{cases} \tag{3.15}$$

Then we sum the first equation with the second:

$$\begin{aligned} S\cos(\chi-\gamma) &= \dot{x}_l cos(\gamma) + \dot{y}_l\sin(\gamma) + \dot{d} \\ \dot{d} &= S\cos(\chi-\gamma) - \dot{x}_l cos(\gamma) - \dot{y}_l\sin(\gamma) \end{aligned} \tag{3.16}$$

Furthermore, $\dot{x}_l$ and $\dot{y}_l$ can be expressed using the dynamic law for the leader defined in equation (3.11):

$$\begin{aligned} \dot{d} &= S\cos(\chi-\gamma) - S_l\cos(\chi_l)cos(\gamma) - S_l\sin(\chi_l)\sin(\gamma) \\ &= S\cos(\chi-\gamma) - S_l\cos(\chi_l+\gamma) \end{aligned} \tag{3.17}$$

In order to get $\dot{\gamma}$ we take the equation, (3.14) and multiply the first equation to $\sin(\gamma)$ and the second equation $\cos(\gamma)$, then we subtract the second with the first:

$$\begin{aligned} S\sin(\chi-\gamma) &= \dot{X}_p\sin(\gamma) - \dot{y}_p\cos(\gamma) + d\dot{\gamma} \\ \dot{\gamma} &= \frac{S}{d}\sin(\chi-\gamma) - \frac{\dot{x}_p}{d}\sin(\gamma) + \frac{\dot{y}_p}{d}\cos(\gamma) \\ \dot{\gamma} &= \frac{S}{d}\sin(\chi-\gamma) - \frac{S_l}{d}\sin(\chi_l-\gamma) \end{aligned} \tag{3.18}$$

Now, we can express the dynamic of the positional error (x_E, y_E), that is the distance of the follower from its formation, in the leader frame as:

$$\begin{cases} x_E = gap_x + d\sin(\gamma - \frac{\pi}{2} - \chi_l) \\ y_E = gap_y + d\cos(\gamma - \frac{\pi}{2} - \chi_l) \end{cases} \tag{3.19}$$

We can now derive this positional error in the leader frame and obtain its dynamic:

$$\begin{aligned} &\begin{cases} \dot{x}_E = \dot{d}\sin(\gamma - \frac{\pi}{2} - \chi_l) + d\cos(\gamma - \frac{\pi}{2} - \chi_l)\dot{\gamma} \\ \dot{y}_E = \dot{d}\cos(\gamma - \frac{\pi}{2} - \chi_l) - d\sin(\gamma - \frac{\pi}{2} - \chi_l) \end{cases} \\ &\begin{cases} \dot{x}_E = Ssin(\chi - \frac{\pi}{2} - \chi_l) - S_l\sin(-\frac{\pi}{2}) \\ \dot{y}_E = Scos(\chi - \frac{\pi}{2} - \chi_l) - S_l\cos(-\frac{\pi}{2}) \end{cases} \\ &\begin{cases} \dot{x}_E = S\sin(\chi - \frac{\pi}{2} - \chi_l) + S_l \\ \dot{y}_E = S\cos(\chi - \frac{\pi}{2} - \chi_l) \end{cases} \end{aligned} \tag{3.20}$$

We can notice that the positional error in the leader frame depends on the difference of the heading between leader and follower, the ground speed of the follower, and the ground speed of the leader as we could intuitively suppose.

3.2.2. Control law design

The vector field generated for the follower is based on some laws that define the desired value given an input (generally a state of the system). For instance, the figure 3.1 shows a vector field of the desired heading χ_d of a fixed-wing UAV in several positions in a 2D plane generated with the law (3.1). Similarly, we can define laws to generate a vector field for a follower, this will take into consideration the position of the leader plus a gap, this new point will be the goal of the control law, moreover, since this point is moving and we need to reach it and achieve stability while the leader is flying in several different path shapes, also other data from the leader will be taken into consideration to build up the control solution. The leader heading will be used to anticipate turns and follow as close as possible the leader movements, the ground speed of the leader is used to reach the point and then keep the position, we can imagine the setpoint as a point in a 1D space moving at the same speed of the leader, the desired speed of the follower should be the same speed of the leader in the case the follower has reached the position of the point. If the follower is after the point, then the expected ground speed should be lower than the ground speed of the point, on the other side, if the follower is behind the point, it has to keep a speed higher than the leader to reach the goal. Given these considerations, we can define S_d:

$$S^d = S_l - S^\infty \frac{2}{\pi} \tan^{-1}(k_x x_E) \tag{3.21}$$

Where S^∞ is considered as the correction magnitude parameter and it defines proportionally the maximum ground speed correction applied to the leader ground speed. k_x is the proportional gain that defines how aggressive is the correction given an error.
If we apply the same logic also to the heading, we have the leader heading and it will be corrected by a parameter that increases or decreases depending on the follower position. We obtain:

$$\chi^d = \chi_l - \chi^\infty \frac{2}{\pi} \tan^{-1}(k_y y_E) \tag{3.22}$$

This will generate a vector field that has a different heading and speed given a position, its visualization can be represented by an arrow in a space that has the length proportional to the desired ground speed.

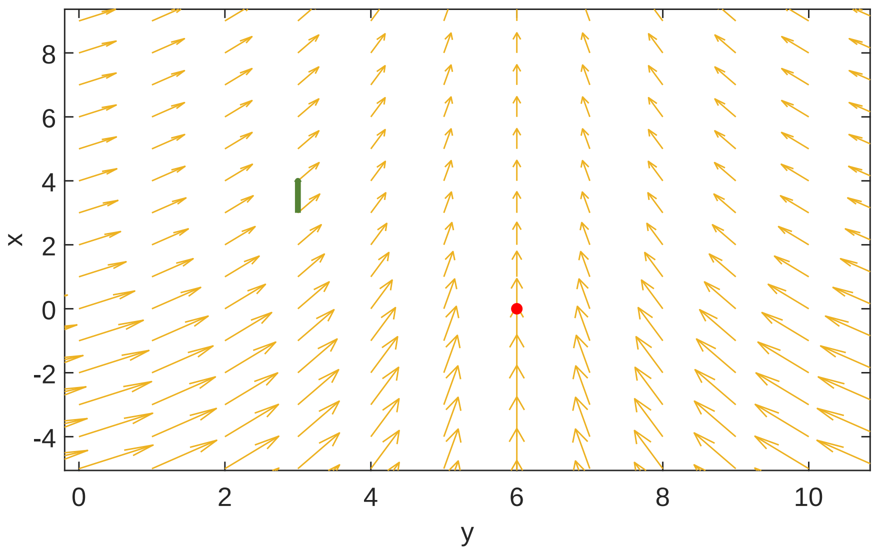

Figure 3.4: This vector field is generated using the equations (3.21) and (3.22), the angle of an arrow indicates the desired heading of the follower in that position, the length of the arrow is proportional to the desired speed. The green arrow is the leader speed position and heading, the red dot represents the position that the follower should reach.

The figure 3.4 shows the previous idea applied in a vector field, as we can see, on the left of the red dot (the goal of the follower), the UAV will correct its position towards the right, if it is on the right if will correct its position on the left. Similarly, if the UAV is above the red dot, its speed is lower than the leader speed, and if it's below the red dot, it will increase its speed to reach the set-points. The magnitude of this correction can be tuned by the parameters χ^{∞}, k_y, S^{∞}, k_x.

Prerequisites to prove the stability

The stability of the system will be proven in this section, this demonstration is based on a candidate Lyapunov function:

$$W = \frac{1}{2}x_E^2 + \frac{1}{2}\rho\tilde{S}^2 + \frac{1}{2}y_E^2 + \frac{1}{2}\rho\tilde{\chi}^2 \tag{3.23}$$

where $\tilde{S}$ is defined as the difference between the ground speed of the follower and its desired ground speed $\tilde{S} = S - S^d$ while $\tilde{\chi}$ is the difference between the follower heading and it is desired heading $\tilde{\chi} = \chi - \chi^d$.
Our S^c and χ^c needs to satisfy this Lyapunov equation (3.23). To find it out, we will do the partial derivative of the Lyapunov function, then solve each block and figure out those two functions.

For the ground speed we take the Lyapunov function and derive it by x_E :

$$\begin{aligned}\dot{W} &= x_E \dot{x}_E + \rho_s \tilde{S}\dot{\tilde{S}} \\ &= x_E(-\tilde{S} - S^{\infty}\frac{2}{\pi}\tan^{-1}(kx_E)) + \rho_s\tilde{S}(\dot{S} - \dot{S}^d) \\ &= -x_E S^{\infty}\frac{2}{\pi}\tan^{-1}(kx_E) + \rho_s\tilde{S}\beta(S^c - S - \frac{1}{\beta}\dot{S}^d - \frac{x_E}{\rho_s\beta})\end{aligned} \tag{3.24}$$

From this step, we can calculate the commanded ground speed such that we obtain a Lyapunov function whose derivative is defined negatively. For instance, we could do:

$$S^c = S + \frac{1}{\beta} + \frac{x_E}{\rho_s\beta}\dot{S}^d - \frac{k}{\beta}\mathrm{sign}(\frac{\tilde{S}}{\epsilon}) \tag{3.25}$$

Thus the equation (3.24) is defined negatively.
Now we can perform the same operation for the commanded heading,here we define a new function that takes into account just the minimization of $\tilde{\chi}$, for instance $w = \frac{1}{2}\tilde{\chi}$ and then we derive it:

$$\begin{aligned}(\frac{1}{2})\rho\dot{\tilde{\chi}}^2 &= \rho\tilde{\chi}\dot{\tilde{\chi}} \\ &= \rho\tilde{\chi}(\dot{\chi} - \dot{\chi}^d) \\ &= \rho\tilde{\chi}\alpha(\chi^c - \chi\frac{1}{\alpha}\dot{\chi}^c) \\ &\rightarrow \chi^c = \chi + \frac{1}{\alpha}\dot{\chi}^d - \frac{k}{\alpha}(\frac{\tilde{\chi}}{\epsilon})\end{aligned} \tag{3.26}$$

Then we can demonstrate the Lyapunov stability by splitting it into two subproblems, first, we will analyze the convergence of ground speed S and the horizontal positional error x_e (in the leader frame) together since their dynamic is directly linked. Then we apply the same logic to the vertical positional error y_e and the heading of the follower χ.

3.2.3. Vertical positional error and heading convergence stability

The vertical positional error y_E and the heading are two dynamics that directly affect each other, thus we will analyze them together and we will use an approach that is quite similar to the one used in [9]. First, starting from the equation (3.23), we simply define the sub problem based on the Lyapunov function w_2:

$$\begin{aligned}\dot{w}_2 &= y_E\dot{y}_E + \rho\dot{\tilde{\chi}}\chi \\ &= Sy_E\sin(\tilde{\chi} + \chi^d - \chi_l) - \rho\frac{k}{\epsilon}\tilde{\chi}^2\end{aligned} \tag{3.27}$$

Then we have to find an upper bound and find for which conditions it is defined negative:

$$\begin{aligned}\dot{w}_2 &= Sy_E\sin(\tilde{\chi} + \chi^d - \chi_l) - \rho\frac{k}{\epsilon}\tilde{\chi}^2 \\ &\leq -\rho\frac{k}{\epsilon}\tilde{\chi}^2 + Sy_E\sin(\hat{\chi}^d) + Sy_E\sin(\hat{\chi}^d + \tilde{\chi}) - \sin(\hat{\chi}^d)\end{aligned} \tag{3.28}$$

Nothing that:

$$\begin{aligned}
&|\sin(\hat{\chi}^d + \tilde{\chi}) - \sin(\hat{\chi}^d)| \\
&= |\sin(\hat{\chi}^d)\cos(\tilde{\chi}) + \cos(\hat{\chi}^d)\sin(\tilde{\chi}) - \sin(\hat{\chi}^d)| \\
&= |\sin(\hat{\chi}^d)(\cos(\tilde{\chi}) - 1) + \cos(\hat{\chi}^d)\sin(\tilde{\chi})| \\
&\leq |\cos(\tilde{\chi} - 1) + \sin(\tilde{\chi})| \\
&\leq 2|\tilde{\chi}|
\end{aligned}$$

we get:

$$\begin{aligned}
\dot{w}_2 &\leq -\rho\frac{k}{\epsilon}\tilde{\chi} + 2Sy_E|\tilde{\chi}| + Sy_E\sin(\chi^d - \chi_l) \\
&\leq -\rho\frac{k}{\epsilon}\tilde{\chi} + 2Sy_E\tilde{\chi} - Sy_E\sin(\chi^\infty\frac{2}{\pi}\tan^{-1}(ky_E) - \frac{\pi}{2})
\end{aligned} \tag{3.29}$$

Then, in order to define when this function is negative, we can distinguish two cases that depend on $\tan^{-1}(ky_E)$, in its saturation region:

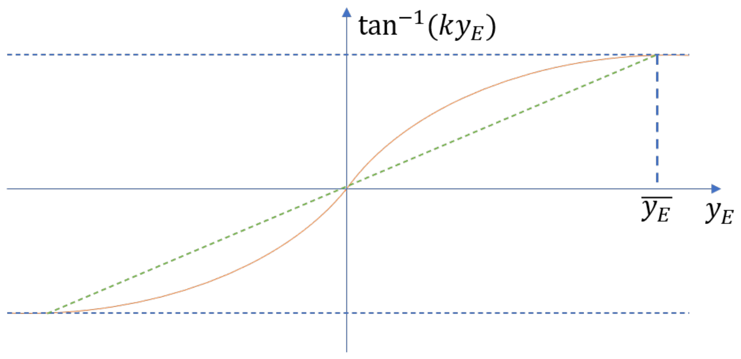

Figure 3.5: $\tan^{-1}(ky_E)$ is in saturation if $y_E > \overline{y_E}$

and we define the function:

$$\phi(y_E) = y_E\sin(\frac{2\chi^\infty}{\pi}\tan^{-1}(ky_E) - \frac{\pi}{2}) \tag{3.30}$$

and we note that $\phi(y_E) \approx (2\chi^\infty k/\pi)y_E^2$ for $ky \longrightarrow 0$ and $\phi(y_E) \approx (\sin\chi^\infty)y_E$ for large value of ky. Consider a new function defined as:

$$\varphi(y_E) = \begin{cases} \frac{2\chi^\infty k}{\mu\pi}y_E^2, & \text{if}|y_E| \leq \overline{y_E} \\ \frac{2\chi^\infty k}{\mu\pi}(2|y_E| - \overline{y_E}), & \text{if}|y_E| > \overline{y_E} \end{cases} \tag{3.31}$$

Then we have to find μ such that $0 < \varphi(y_E) \leq \phi(y_E)$, note that both are symmetric functions in y_E, therefore, we will restrict our attention to $y_E \geq 0$. To show that $\varphi(y_E) \leq$

$\phi(y_E)$ we will use the theorem: Given two functions f and g that satisfy $f(0) = g(0)$ and $f'(y) \leq g'(y)$ for $y \geq 0$, then $f(y) \leq g(y)$ for $y \geq 0$.
Given that, consider $0 \leq y_E \leq \overline{y_E}$:

$$\begin{aligned}\phi'(y_E) &= \sin(\frac{2\chi^\infty}{\pi}\tan^{-1}(ky_E) - \frac{\pi}{2}) + \frac{2k\chi^\infty}{\pi}y_E\left[\frac{cos(\frac{2\chi^\infty\tan^{-1}(ky_E)}{\pi})}{1+(ky_E)^2}\right]\\ &\geq \frac{4\chi^\infty k}{\pi}y_E\left[\frac{1}{2}\frac{cos(\frac{2\chi^\infty\tan^{-1}(ky_E)}{\pi})}{1+(ky_E)^2}\right]\\ &\geq \frac{4\chi^\infty k}{\pi}y_E\left[\frac{1}{2}\frac{cos(\frac{2\chi^\infty\tan^{-1}(k\overline{y_E})}{\pi})}{1+(k\overline{y_E})^2}\right]\\ &\geq \frac{4\chi^\infty k}{\pi}y_E\\ &= \varphi'(y_E)\end{aligned} \tag{3.32}$$

if we consider that:

$$\mu \geq \frac{2(1+(k\overline{y_E})^2)}{cos(\frac{2\chi^\infty}{\pi}\tan^{-1}(k\overline{y_E}))} \tag{3.33}$$

On the other side, we have to consider $y_E \geq \overline{y_E}$, thus ϕ become:

$$\begin{aligned}\phi(y_E) &= y\sin(\frac{2\chi^\infty}{\pi}\tan^{-1}(ky_E))\\ &\geq y\sin(\frac{2\chi^\infty}{\pi}\tan^{-1}(k\overline{y_E}))\end{aligned} \tag{3.34}$$

this implies that $\phi(y_E) \geq \varphi(y_E)$ if:

$$\mu \geq \frac{4\chi^\infty k\overline{y_E}}{\pi\sin(\frac{2\chi^\infty}{\pi}\tan^{-1}(k\overline{y_E}))} \tag{3.35}$$

then $\varphi(y_E) \leq \phi(y_E)$ which implies:

$$\dot{W}_2 \leq -\frac{\rho k}{\epsilon}\tilde{\chi}^2 + 2V_g|y_E||\tilde{\chi}| - V_g\varphi(y_E). \tag{3.36}$$

Therefore for $|y_E| \leq \overline{y_E}$:

$$\dot{w}_2 \leq -S(\begin{pmatrix}|\tilde{\chi}| & |y_E|\end{pmatrix})\begin{pmatrix}\frac{\rho k}{\delta S} & -1\\ -1 & \frac{2\chi^\infty k}{\mu\pi}\end{pmatrix}\begin{pmatrix}|\tilde{\chi}|\\ |y_E|\end{pmatrix} \tag{3.37}$$

and system converge if the following equation is satisfied:

$$\frac{\rho k 2\chi^\infty k}{\epsilon S\mu\pi} > 1 \tag{3.38}$$

else if $|y_E| \geq \overline{y_E}$ (is in the saturation level) the convergence is satisfied when:

$$\dot{w}_2 \leq 2Sy_E(\epsilon - \frac{\chi^\infty k\overline{y_E}|y_E|}{\mu\pi}) \rightarrow \frac{\chi^\infty k\overline{y_E}}{\mu\epsilon\pi} > 1 \tag{3.39}$$

These results can be interpreted also physically, indeed, in the equation (3.38) the ground is at the denominator, thus if the speed is too high, the plane will not reach a convergence (physically its harder also to turn), while the k represents physically the magnitude of the correction, if its too small, the system can not converge.

3.2.4. HORIZONTAL POSITIONAL ERROR AND GROUND SPEED STABILITY

The horizontal error x_e is directly affected by the ground speed difference between a leader and follower: if the ground speed of the leader is higher than the ground speed of the follower $S_l > S$, or lower, then $|x_e|$ will increase. We can define the difference between the follower ground speed and the desired as $\tilde{S}$:

$$\tilde{S} = S - S_d = S - S_l - S^\infty \frac{2}{\pi}\tan^{-1}(kx_E) \tag{3.40}$$

Now it is possible to provide a Lyapunov function of the problem and prove that it is defined negative, by taking into consideration the dynamics previously defined, and the equation (3.23) we can define another sub-problem w_1:

$$\begin{aligned} \dot{w}_1 &= x_E\dot{x}_E + \rho\tilde{S}\dot{\tilde{S}} \\ &= Sx_E\sin(\tilde{\chi} + \chi^d - \chi_l - \frac{\pi}{2}) - \rho\frac{k}{\epsilon}\tilde{S}^2 \end{aligned} \tag{3.41}$$

Then is possible to prove that the Lyapunov function is defined negative by finding a function that is its upper bound limit defined negative.

$$\begin{aligned} \dot{w}_1 &= Sx_E\sin(\tilde{\chi} + \chi^d - \chi_l - \frac{pi}{2}) - \rho\frac{k}{\epsilon}\tilde{S}^2 \\ &\leq -\rho\frac{k}{\epsilon}\tilde{S}^2 + Sx_E\sin(\hat{\chi}^d) - Sx_E\sin(\hat{\chi}^d + \tilde{\chi}) - \sin(\hat{\chi}^d) + S_Lx_E \\ &\leq -\rho\frac{k}{\epsilon}\tilde{S}^2 + 2Sx_E\tilde{\chi} - Sx_E\sin(\chi^d - \chi_l - \frac{\pi}{2}) + S_lx_E \\ &\leq -\rho\frac{k}{\epsilon}\tilde{S}^2 + 2Sx_E\tilde{\chi} - Sx_E\sin(\chi^\infty\frac{2}{\pi}\tan^{-1}(ky_E) - \frac{\pi}{2}) + S_lx_E \\ &\leq -Sx_E\cos(\tilde{\chi} + \chi^d - \chi_l) + S_lx_E - \rho\frac{k}{\epsilon}\tilde{S}^2 \\ &\leq x_E(S_l - \cos(\tilde{\chi} + \chi^d - \chi_l)) - \rho\frac{k}{\epsilon}\tilde{S}^2 \\ &\leq x_E(S_l - \cos(\tilde{\chi} + \chi^\infty\frac{2}{\pi}\tan^{-1}(ky_E))) - \rho\frac{k}{\epsilon}\tilde{S}^2 \end{aligned} \tag{3.42}$$

Then, by taking the equation (3.40) and applying to it the same logic used in the previous section to find the saturation of the correction:

$$\begin{aligned}\tilde{S} &= S - S_d \\ &\leq -\tilde{S} - S^{\infty} x_E \\ &\leq -\tilde{S} - S^{\infty} \frac{2}{\pi} \tan^{-1}(k x_E) \\ &\leq -\tilde{S} - \frac{S^{\infty}}{\mu} x_E\end{aligned} \tag{3.43}$$

Moreover, if we use the results from w_2, by considering $y_E \to 0$ and $\tilde{\chi} \to 0$, we have that:

$$\begin{aligned}S x_E \sin(-\frac{\pi}{2}) + S_l x_E \approx (S_l - S) X_e \\ \leq -(\tilde{S} + \frac{S^{\infty}}{\mu} x_E) x_E\end{aligned} \tag{3.44}$$

Now we put this results in the equation (3.42):

$$\dot{w}_1 \leq \rho \frac{k}{\epsilon} \tilde{S}^2 + 2 S x_E \tilde{X} - (\tilde{S} + \frac{S^{\infty}}{\mu} x_E) x_E \tag{3.45}$$

for $x_E \leq \overline{x_E}$:

$$\dot{w}_1 \leq -\left((\tilde{S} \quad x_E)\right)) \begin{pmatrix} \frac{\rho k}{\epsilon} & \frac{1}{2} \\ \frac{1}{2} & \frac{S^{\infty}}{\mu} \end{pmatrix} \begin{pmatrix} \tilde{S} \\ x_E \end{pmatrix} \tag{3.46}$$

The function is defined negative if the following equation is satisfied:

$$\frac{4\rho k}{\mu \epsilon} S^{\infty} > 1 \tag{3.47}$$

while for $x_E > \overline{x_E}$:

$$\begin{aligned}\dot{w}_1 &\leq -\frac{\rho k \tilde{S}^2}{\epsilon} + (-S^2 - S^{\infty} \frac{2}{\pi} \epsilon) x_E \\ &\leq -\frac{\rho k \tilde{S}^2}{\epsilon} - \tilde{S} x_E - \frac{\epsilon}{4\rho k} x_E^2 + x_E (\frac{\epsilon}{4\rho k} x_E - S^{\infty} \frac{2}{\pi} \epsilon) \\ &\leq -(\sqrt{\frac{\rho k \tilde{S}}{\epsilon}} \tilde{S}^2 + \frac{1}{2} \sqrt{\frac{\epsilon}{4\rho k}} x_E)^2 + (\frac{\epsilon}{4\rho k} x_E - S^{\infty} \frac{2}{\pi} \epsilon)\end{aligned} \tag{3.48}$$

is negative defined if:

$$\frac{8 S^{\infty} \rho k \epsilon}{\pi \epsilon} > 1 \tag{3.49}$$

3.3. Resume

In this chapter we have introduced the vector field solution for path following, a similar approach is then extended and adapted to be used for formation control. We first defined the

desired dynamic of the UAV, then we used the dynamic model of the UAV and the formation control goals to calculate the commanded heading and ground speed.
In the end, we discussed its stability by dividing the problem into two subproblems, we found that the system is stable by some constrains on the parameter and that the ground speed of the leader shall be less than the maximum speed of the follower.

4

Implementation of the solution and its simulation

In this section, we implement the system by doing more precise considerations on how the whole solution was implemented in an embedded system. The first step is the definition of a software architecture that abstracts all the agents in the system and the communication between them. Once the system is implemented, we will analyze the robustness by performing some test software in the loop and hardware in the loop, also by introducing some faults and noises to check whether the system can handle real-case scenarios. In the end, the performance of the system will be evaluated with different paths and windy conditions.

4.1. Implementation

A software implementation of the proposed solution is essential to evaluate the performances and to validate the theoretical assumptions, the vector field for formation control will be implemented such that it is possible to run it also in a simulation environment and bring it easily on a real UAV. Thus, it's crucial to design every aspect of the simulating system from the beginning to avoid double work and unmodeled differences from a real system, like computational limits or communication delays.
Since the focus of this thesis is not on the low-level controller, neither on the UAV modeling or state estimation we decided to use PX4 and keep its low-level flight stack, while we changed the high-level mission control algorithms.

4.1.1. Autopilot architecture for formation control

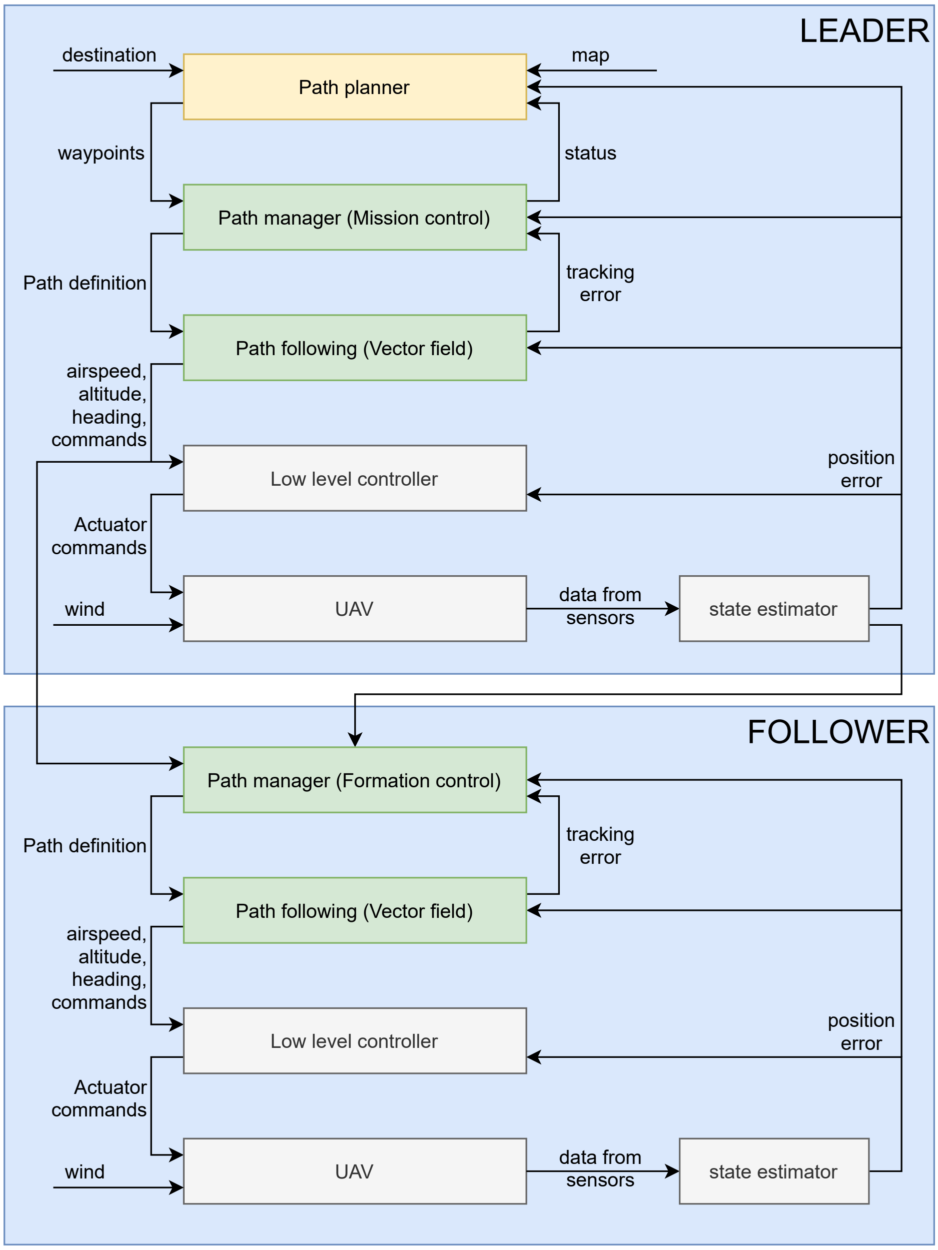

Figure 4.1: This autopilot architecture is based on [9] and it is modified to integrate the follower autopilot architecture

The autopilot architecture defined in [9] took into consideration just one agent of the system, now we have multiple agents that exchange data to reach a goal.
Thus, we modified the autopilot structure such that the follower receives the position, attitude, and heading commands from the vector field calculator and the state estimator. From figure (4.1) is clear that the follower does not need a path planner, since the path is already planned by the leader based on the mission. While the path manager and the path following layers are different between the leader and the follower since the path manager of the follower has to take into account the data coming from the leader instead of the waypoints and the path following layer will generate a vector field for formation control instead of a normal vector field. The lower layers are the same for both the agent type.
The follower block can be replicated with multiple instances in order to have multiple followers.

4.1.2. Software architecture

Before the design of the software architecture, we defined the agents of the system, how they communicate, and which information they need, then all the agents are structured in layers, interconnected by interfaces. The interfaces can be data busses or electronics interconnections. The figure 4.2 shows a logical view of our system.

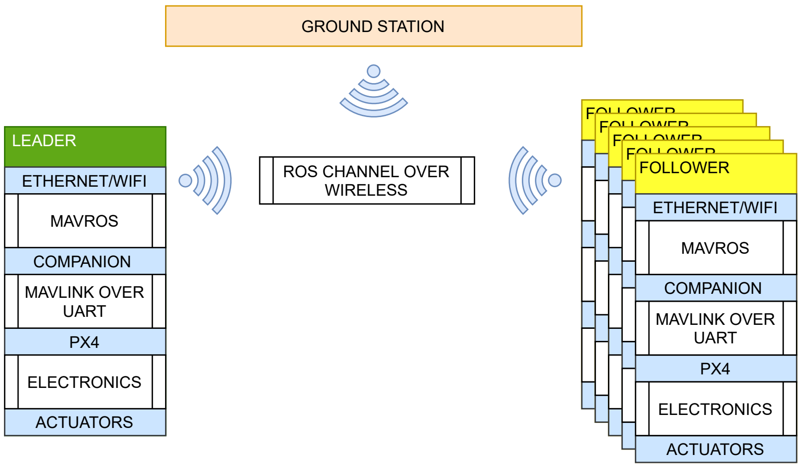

Figure 4.2: A logical view of the software architecture used and the communication buses, there are three main type of components: a ground station, leader and follower.

Leader and follower have the same layered structure, the lowest layer is made by physical actuators and sensors, such as step-by-step motors for the mobile panels, brush-less motors to generate lift, and sensor as described in section 2.2. The actuators and sensors layer is connected to the UAV controller PX4 through some electronics.
The PX4 layer has all the interfaces to control the motors (using a PWM-based communication) and the algorithms for sensors fusion, moreover it has all the low-level control loop to control the attitude of the plane, all the gray boxes in 2.3 are included in the PX4 layer.
The PX4 firmware is developed to work on a restricted number of devices, those devices are

usually embedded micro-controllers that have a limited amount of resources, thus, in order to have a wide possibility of implementation, we interconnected the PX4 with companion hardware that implements all the high-level functions. Those two layers are interconnected by a serial RS232 over USB, this bus uses the MAVLink protocol to communicate between the companion hardware and the PX4.
MAVLink is a standard protocol created for the communication within the devices on the UAV and external systems, it is based on several messages with a predefined structure.
The companion hardware used in our simulation is a Raspberry PI 3 B+, a single-board computer equipped with wireless LAN connectivity, a Quad-core 1.2GHz CPU, and 1GB RAM allow to implement complex calculation and handle communication protocols easily. The companion device has Linux as the operating system and the communication stack and the algorithm used to calculate the vector field for the leader and to calculate the vector field for formation control for the followers.
The communication stack used by the companion hardware is ROS (Robot operating system) introduced in the section 2.3.1 and it uses the LAN network over RF antennas to send and receive data between all the other system components.
The other main component is the ground station, it is used to visualize the mission plan, add or modify missions, or configure the parameters of the flight controllers. In our simulation, we used QGroundControl, a project made by Linux Foundation that works with all the UAVs that use the MAVLink protocol.

4.1.3. From Vector Field to UAV control

PX4 contains several flight modes that define how the autopilot behaves given external command. Since the focus of this thesis is just about path following and formation control, we use the mode "altitude": the internal control loops of PX4 calculate the pitch value to keep a given altitude, then it reads the throttle, roll, and yaw values from an external system to control the heading and the speed. Moreover, is important to notice that when the PX4 is in "altitude" mode, a command to set a different roll angle will also change the yaw, since the PX4 keeps the coordinated flight (flying without side-slip), so there is no need to control the yaw directly.

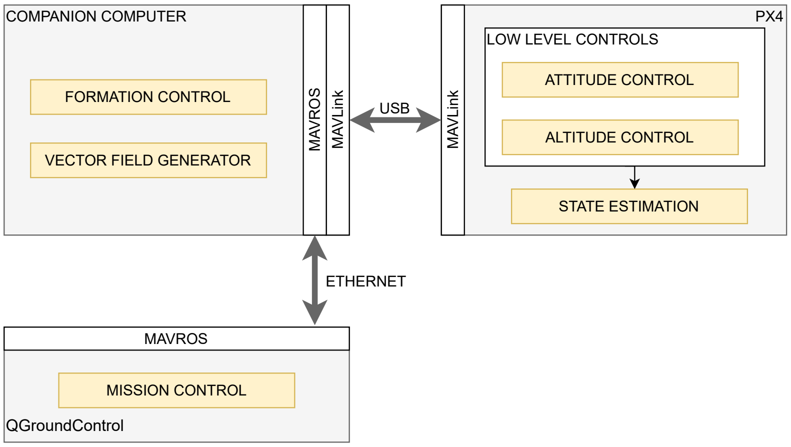

Figure 4.3: Logic representation of the software used for the control of the UAV

To recap, the PX4 uses its low-level controller to stabilize the flight and control the actuators moreover it reads the sensors and elaborates the data for the state estimation, used to feed the low-level controllers. The PX4 receives from the companion hardware the roll and thrust value, that will be used to change the heading and the speed of the UAV. The companion hardware receives the data from the state estimation in PX4 to calculate the vector field for path tracking or for path following. Furthermore, the companion hardware can communicate with the ground control station to get and set updates about the mission.

4.1.4. ROS node structure

3 main nodes were developed to implement this solution: vector field, is the node used by the leader to generate a vector field for path following, using the law proposed in [9], these nodes receive the data through MAVROS about the UAV and its location, heading, attitude, thus it generates the vector field. The formation control node is used to generate a vector field for formation control, it receives the data from the UAV MAVROS node and the leader UAV node, then it computes the heading and speed, that is sent to the joystick simulator node, this node is made by 2 simple proportional hold control loop used to transform the heading value to a RCcommand that is sent to the follower UAV by MAVROS.

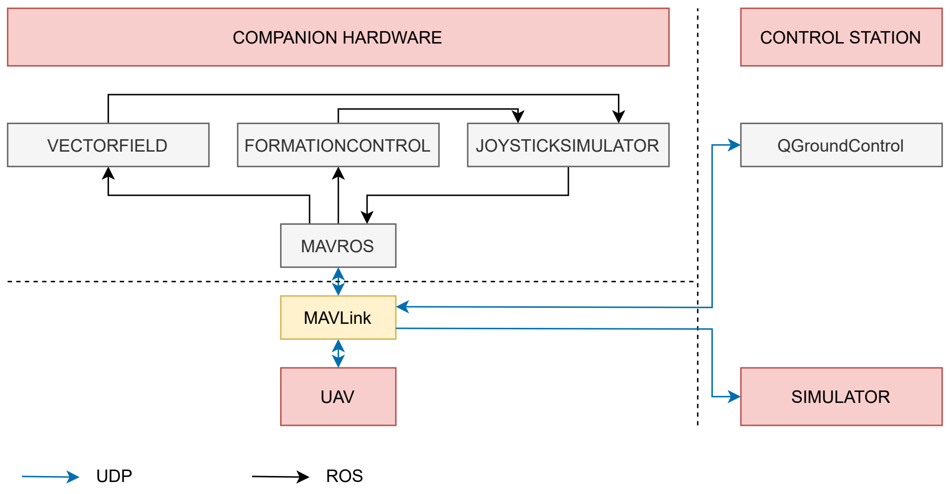

Figure 4.4: The structure of the nodes, the dependencies and the udp connection links

Some of the components are interconnected through a UDP connection like the simulator and the QGroundControl, this is to improve the speed of the link, since ROS is TCP based and it introduces a lot of overhead, that decreases the throughput, thus the overall time to reply of the system.

4.2. Implementation of the system and test in a simulation environment

The goal of the simulation is to test the proposed solution with a computer-modeled vehicle in a simulated world, a simulation decreases the costs of development, since it allows to test rapidly and almost cost-less a solution. It is crucial to have a proper working code and control strategy before actually flying with a real UAV, to avoid crashes and errors that could lead to a fatal situation.
The solution shall be able to behave like a real environment, the plane should have a realistic dynamic, the world should also be simulated with the windy situation and magnetic interference that could happen in a real situation.
The simulation of the physical world is made possible by Gazebo Simulator, an open-source 3D simulator that integrates the Open Dynamic Engine, a physics engine written in C/C++ with rigid body dynamics, aerodynamics engine, and quite optimized simulation of actuators and sensors. Thus we can simulate a different noise for each of the sensors to test the solution for several cases.
Based on the focus of the tests, there are two main possibilities for testing: Software In the Loop (SITL) and Hardware In the Loop (HITL).

4.2.1. The software in the loop

The software in the loop simulation is used to test the solution in a simulated environment using a mathematical model of the plant. This type of test allows to solve most of the problems that can be generated by human errors during the development, moreover, is possible to test also the control loop solution and solve errors related to the dynamics of the system. More realistic is the mathematical model more will be the problems solved during the simulation phase. Our algorithm was developed and tested in SITL mode, following the guidelines available on PX4 documentation and based on [27].

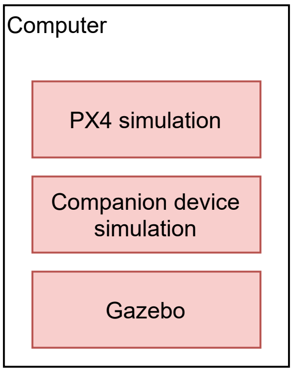

Figure 4.5: High level view of a software in the loop simulation

4.2.2. Hardware in the loop

Hardware in the loop is a similar approach, but the code is executed on physical hardware that is interfaced to a simulated system.
The advantage of this approach is that its possible to verify the compatibility of the implementation with the hardware since there are two types of incompatibilities that could lead to a poor performance of the system: the solution is complex by space and computation or there are behavior non modeled in the simulations.
In the first case, it is possible to perform an initial evaluation of the complexity in the algorithm and estimate the resources needed to execute all the calculation in a reliable amount of time and space, in our case, we are using a Raspberry PI that has a quite lot of computation and the algorithm for formation control has a complexity of O(1) and it takes a few memory locations, thus the required computational resource can be reduced. Secondly, some dynamics could not be present in the SITL testing, like the time delays introduced in the system for the communication between the companion hardware and the microcontroller running PX4.
Moreover, it is possible to estimate this time delay, since the companion hardware and the PX4 board communicates using the RS-232 protocol (over UART) with a baudrate of 115200, this means that the protocol transmits 115200 bytes in one second, on this, we have to introduce the computational time to decode the frame and extract the message from the MAVLink protocol, this time is far way smaller than 1 millisecond on an STM32F7 Microcontroller as used by most of the PX4 boards. A Mavlink frame is made by up to 279 bytes including header and CRC, thus it will take 2.5 milliseconds to transmit the frame

over UART and 1 millisecond to decode it. By these assumptions and measurements, the companion hardware and the PX4 can exchange updates with a frequency up to 200Hz.

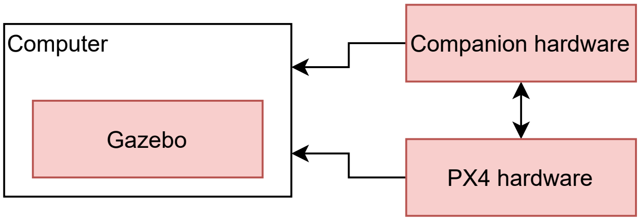

Figure 4.6: High level view of a hardware in the loop simulation

4.2.3. Resume

In this section, we discussed how we implemented the proposed solution in an embedded platform by using state of the art communication frameworks and UAV control frameworks. At first, we defined a new autopilot architecture for the followers, designed to satisfy the requirements of robustness and scalability.
This new architecture is then implemented using the ROS framework for communication between UAVs. In the end, we validated the system by software in the loop simulations.

5 Performance evaluation

The original plan of the thesis was to evaluate the performance of the system with a real flight test in a laboratory located in China, but due to some travel restrictions in 2020, we evaluated the system in a SITL simulation environment.
In this chapter, we will test the system with different path types, and with several wind conditions and sensor disturbance.

5.1. Simulations without wind

In this section, we test the system for different shapes and situations in the simulation environment. The weather is considered to be still and windless. Some noises are introduced into the system using the plugins provided by Gazebo. The noises introduced on GPS, magnetometer, accelerometer and gyroscope are based on a random-walk dynamic. During this test set, the wind force is set to zero. All the parameters that define the world, wind, weather, aircraft structure, and dynamics used in the simulation are defined in the file "plane.sdf" inside the sitl_gazebo folder.

5.1.1. Straight trajectory

In this test, the leader is flying on a straight path while the follower starts from a perpendicular position at a distance of 300 meters from the leader's path. The flying speed of the leader is 18 meters/s, while the follower's maximum speed is 21 meters/s.

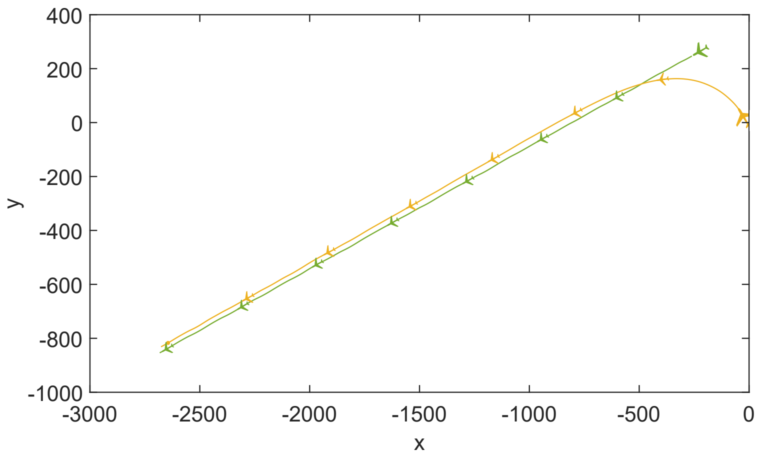

Figure 5.1: The leader plane (in green) is flying on a straight line path, while a follower plane (in yellow) reaches its position in the formation control. The plot measure unit are relative to the simulation environment, but can be considered as meters.

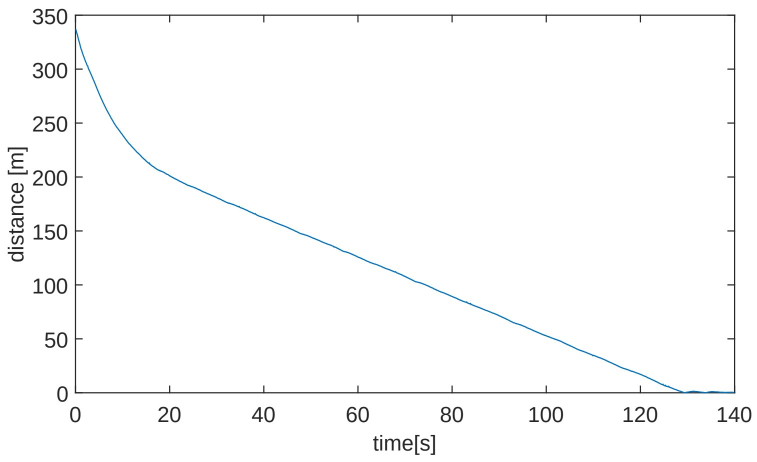

Figure 5.2: The plot shows the distance from the follower and the setpoint to reach the formation.

5.1.2. Circular trajectory

The circular trajectory is harder for formation control since it consists of keeping a formation while the two planes are continuously chaining their heading. Moreover, during a turn, to avoid side-slips the plane has to control its rolling angle and the back rudders to change the yaw and pitch angles. Also in this simulation, the speed of the leader is capped at 18m/s while the speed of the follower is upper bounded at 21m/s.

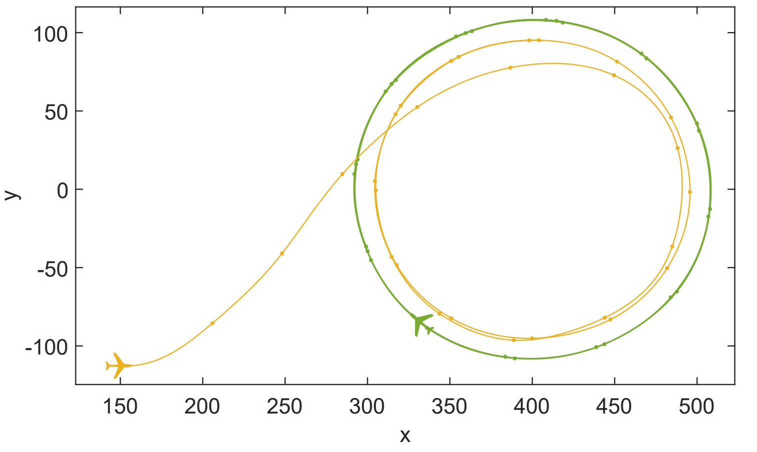

Figure 5.3: The follower plane (in yellow) reaches the trajectory of the leader (in green), then it starts to get aligned with it once it is inside the boundaries.

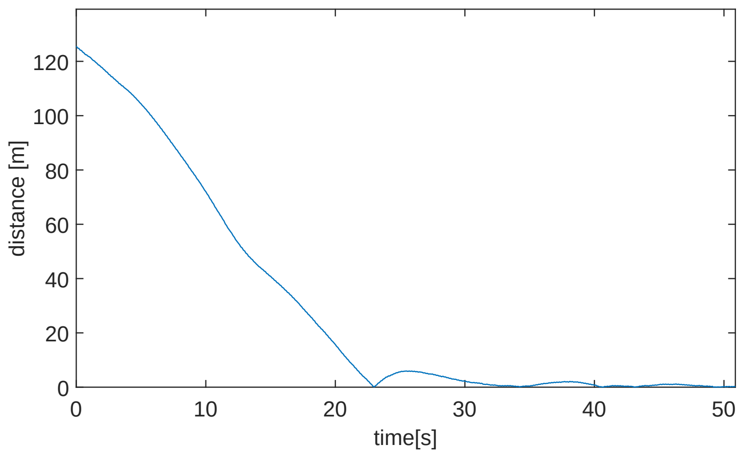

Figure 5.4: The distance error of the follower from its position to reach the formation fly

here the follower has to perform a continuous angle correction, and since the gap makes the orbit of the follower smaller than the orbit of the leader, the roll angle of the follower has to be higher than the leader. A tighter angle of turn makes the control more complex since also the speed must be carefully handled to keep the formation.

5.1.3. Mixed path with multiple followers

In this experiment, we run 4 followers and a leader in a formation fly and check how they keep the formation along a mixed shape path. This path is sent to the leader as a list of waypoints, once the leader reaches 90% of the path, it changes the waypoint and aims to the next in the list.

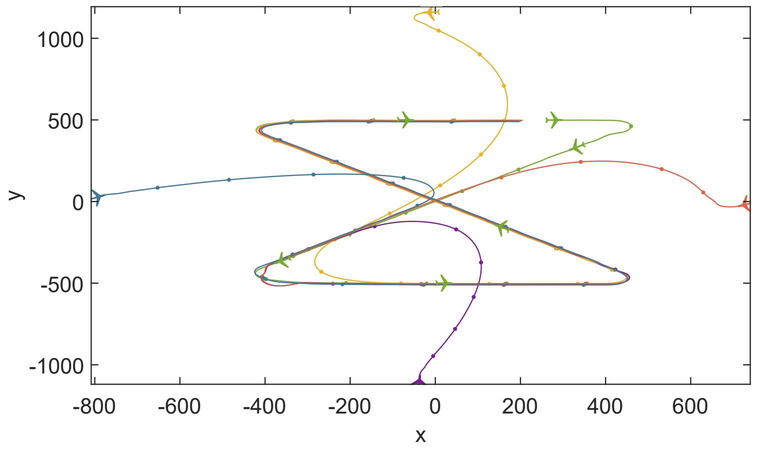

Figure 5.5: The follower plane (in yellow) follows the leader (in green), the follower lag behind the leader and performs some over-correction of the trajectory due to the harsh wind conditions.

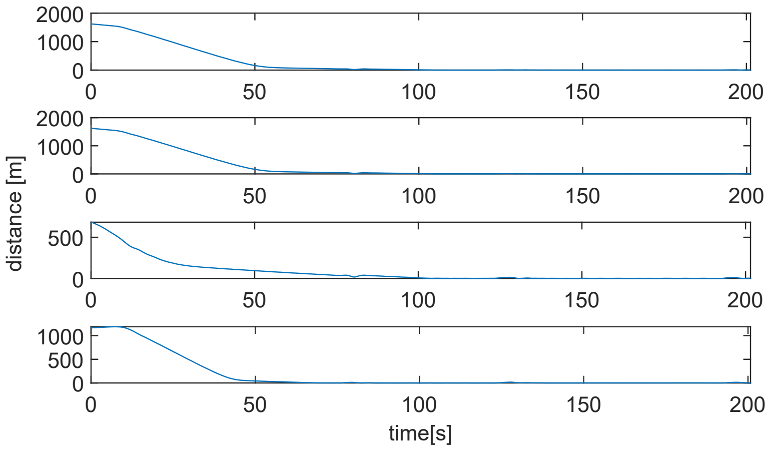

Figure 5.6: The figure shows the distance between the follower and its setpoint to keep the formation.

5.2. Strong wind tests

A quite strong wind is simulated in Gazebo using the wind plugin, the noise is generated using a random walk model and the mean wind speed is approximate of 10m/s.
This is a corner case for the UAV since they can barely fly straight against the wind.

5.2.1. Straight trajectory

In this simulation, the speed of the leader is the same as during the experiment without wind at 18m/s, while the follower speed has the same maximum speed of 21m/s, but due to the windy condition, it can barely reach 20m/s.

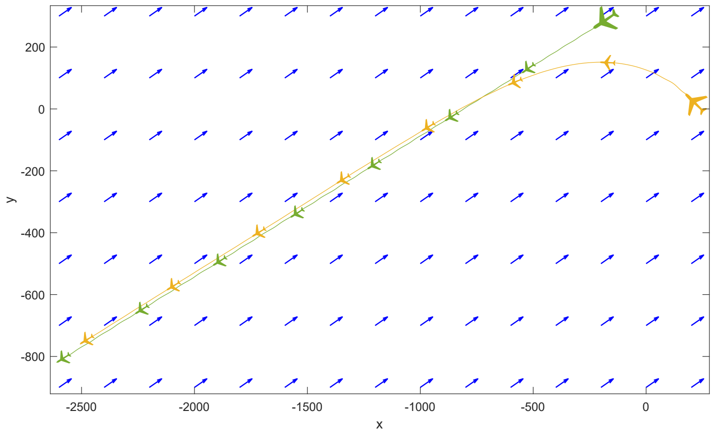

Figure 5.7: The leader plane (in green) is flying on a straight line path, while a follower plane (in yellow) reaches its position in the formation control. The plot measure unit are relative to the simulation environment, but can be considered as meters.

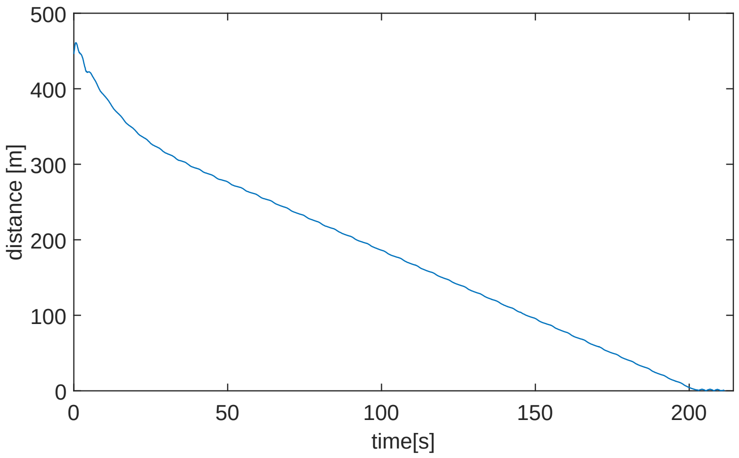

Figure 5.8: The plot shows the distance from the follower and the setpoint to reach the formation.

From the simulation, we can understand that the system can still reach the goal, but in a higher amount of time. A solution to decrease this time is bidirectional communication between leader and follower, such that the leader can know when a follower is not able to catch up with the leader. Thus the leader can decrease its speed or change its ground track to simplify the convergence with the leader.

5.2.2. Circular trajectory

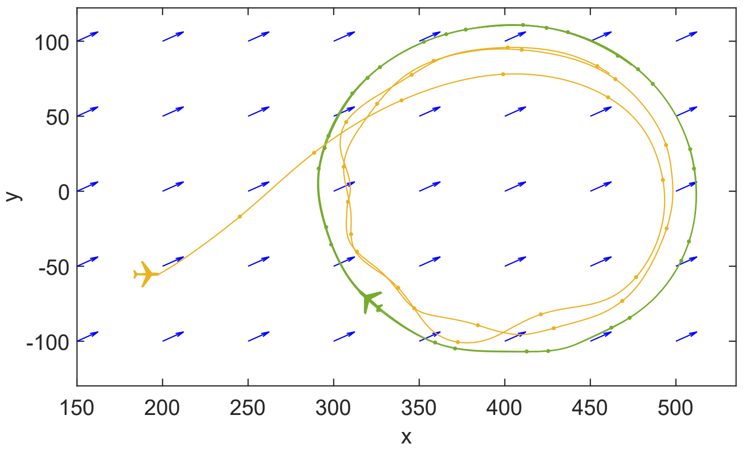

Figure 5.9: The follower plane (in yellow) reaches the trajectory of the leader (in green), then it starts to get aligned with it once it is inside the boundaries. The blue arrows are the wind direction. The path made by the follower is highly irregular in the bottom right part.

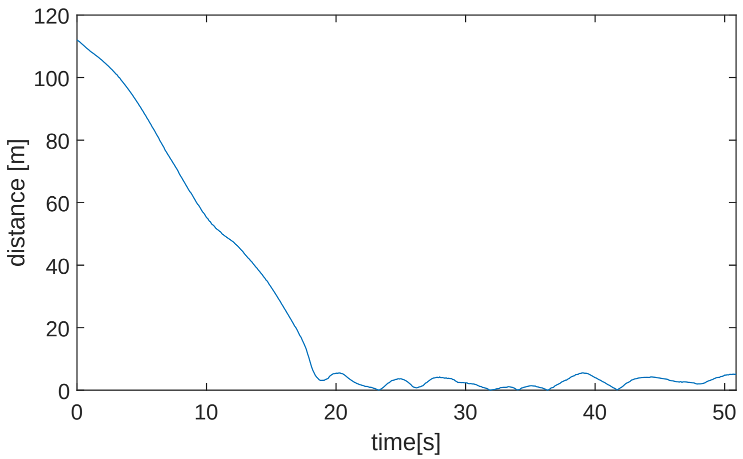

Figure 5.10: The plot shows the distance between the follower and the leader during a strong wind.

In this situation, the follower has a very poor performance. To analyze why this happened, we checked the dynamic of the UAVs in the simulation. The follower, in yellow in the figure 5.9shows that it has a lower radius, thus it has to roll more than the leader. During a roll to perform a left turn, if the wind hits the wings from the right, it will push the plane to up-left, changing its trajectory and increasing its altitude, thus the PX4 altitude correction algorithm will decrease the speed of the plane to recover the correct altitude, this decrease of speed reduces the turn angle and make the plane exit from the trajectory. This is what happens when the plane is on the south-west of the circle. On the other side, the effect is less visible on the north-east of the circle since the wind will hit the plane on the upper side, but this can be more easily corrected just by increasing a bit its speed to maintain the altitude and adjust the turn angle.

5.2.3. Mixed path

In this experiment, we run multiple followers and a leader in formation fly and check how they keep it along a mixed shape path with a strong wind. We can see that the followers can keep track but they lose the formation shape during turns since the planes have different turn radius and they are subjected to different forces caused by the wind. Moreover, the followers keep a good trajectory till the flying path is parallel to the x-axis, but if it flies towards the airflow or its perpendicular, then the follower is a lot more subject to errors.

5.2.4. Mixed path with multiple followers

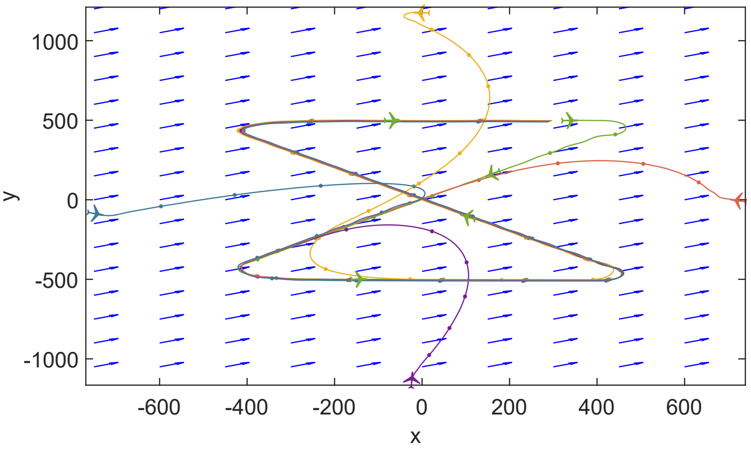

Figure 5.11: The follower plane (in yellow) follows the leader (in green), the follower lag behind the leader and performs some over-correction of the trajectory due to the harsh wind conditions.

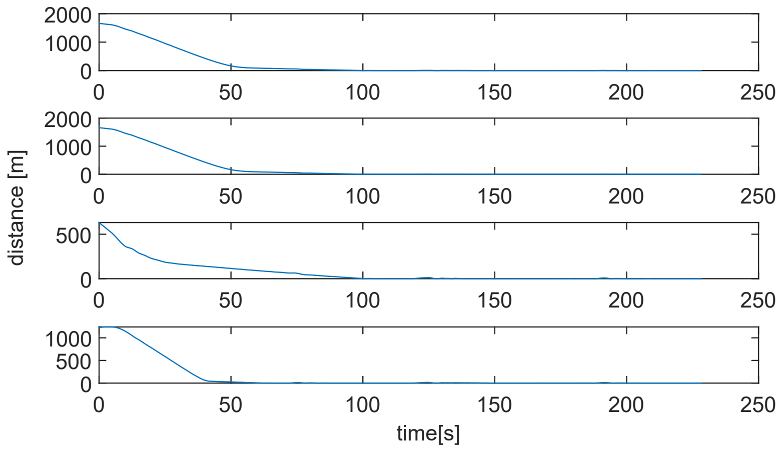

Figure 5.12: The figure shows the distance between the follower and its setpoint to keep the formation.

5.3. Performance comparison with the unicycle-type approach

We evaluated the performances of our solution with a solution proposed [28] that we will refer to as Unicycle-type Approach or UT in the following section. The two algorithms are implemented in MATLAB and a simple dynamic is used (3.14). The UT solution is a tracking control for a group of unicycle mobile robots. Based on the actual UAV coordinates, reference velocity of individual UAV are calculated in real-time such as to ensure collision-free movement.

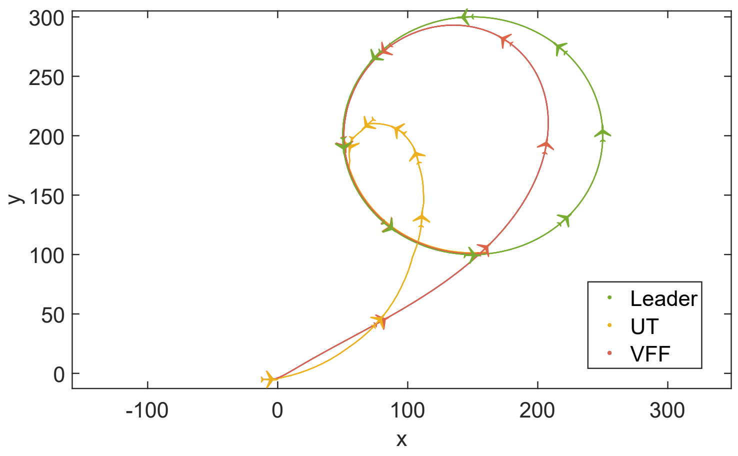

Figure 5.13: The leader (in green) has a circular path in green and starts from (150,100), the two followers in yellow and red starts from a different position (0,0).

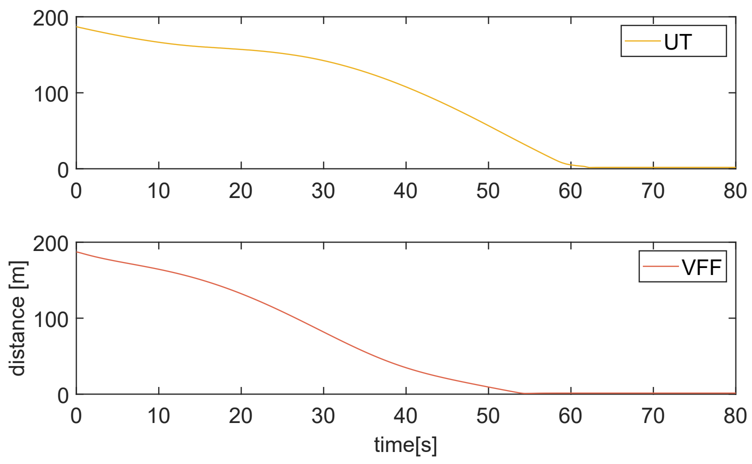

Figure 5.14: The figure shows the distance between the follower and its setpoint.

The UT algorithm physically is highly affected by the leader heading, indeed, in the beginning, the UT follower is heading differently than the VFF follower, which is heading towards the leader, while the UT follower towards the same as the leader. This behavior is more visible in this second experiment with straight lines:

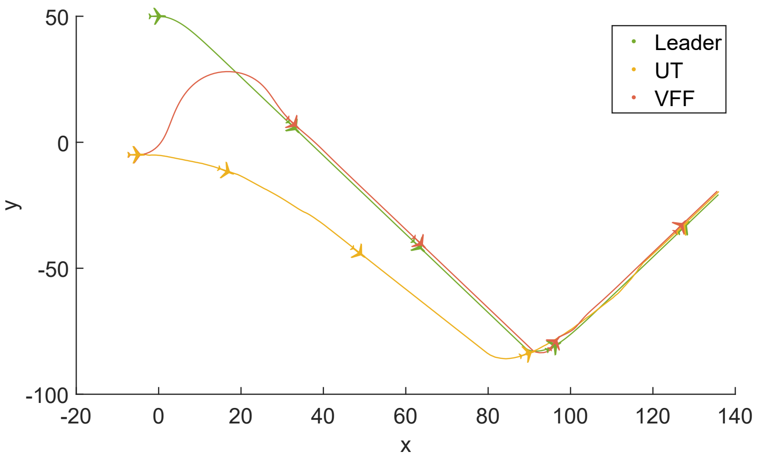

Figure 5.15: The leader (in green) has a straight path in green and starts from (0,50), the two followers in yellow and red starts from a different position (0,0).

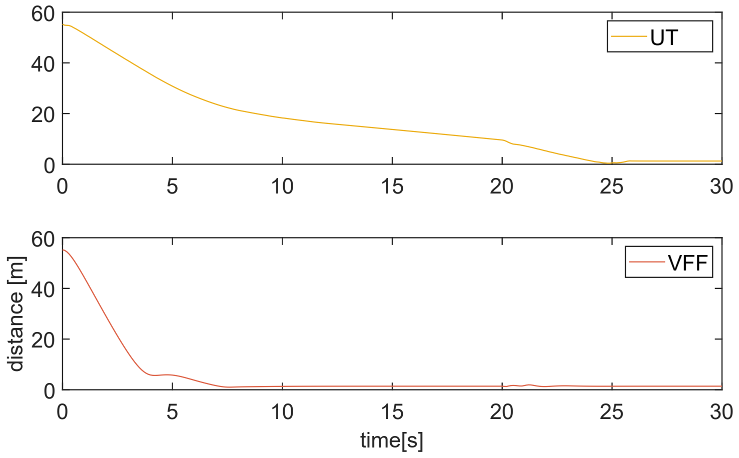

Figure 5.16: The figure shows the distance between the follower and its setpoint.

In this experiment, the leader follows two straight lines. The VFF follower establishes the formation quite fast after a few seconds, flying towards the leader, and once inside its boundary it aligns up. On the other side, the UT follower flies parallel to the leader imitating his heading and slowly getting closer to the formation, when the leader performs a turn, the UT follower align with the formation. This difference is due to the approaches used to construct the control law.
In VFF, the follower navigate towards the leader if the distance is above a threshold, but this is not the case in the UT approach, since that algorithm is developed to follow a reference path/point, it does not take into consideration the situation in which it is too far from that reference, but it starts to "imitate" the reference agent from the beginning and slowly corrects the error. To recap the performance during the transient state, we run several experiments and collect the data in a table.

	UT	VFF
Exp. 1	61.1s	54.4s
Exp. 2	24.1s	7.2s
Average 20 exp.	35.5s	27s

As we can see, the VFF controller reaches the setpoint of formation faster than UT, even during the 20 experiments, we observed that the VFF was faster or equal to the UT controller.

5.3.1. steady-state error

In order to study the steady state error performance of the two control solution, we set up an experiment using a simulation environment in windy conditions. The leader flies a straight line and the two UAV have reached their set-point value.
From here we recorded several minutes of flight to estimate an average performances.

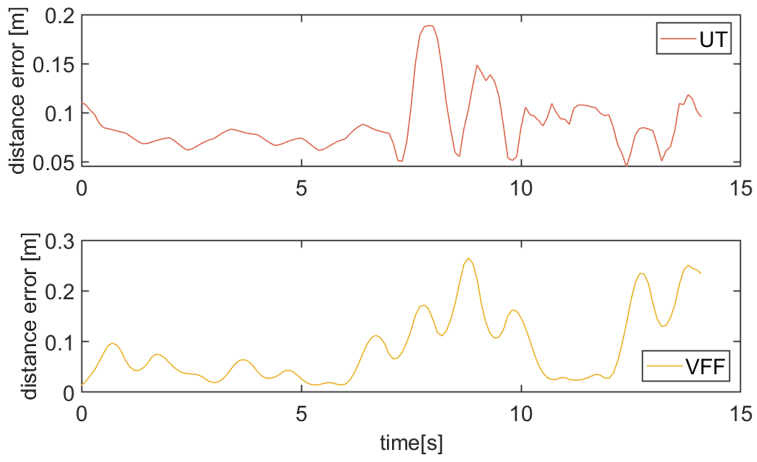

Figure 5.17: The figure shows a 10s section of the distance between the follower and its setpoint for the UT and VFF controllers.

Overall, the UT controller rejects slightly better the error than VFF. The UT controller achieved an average error of 12cm with a standard deviation of 6cm, while the follower using the VFF law reached an average error of 14cm with a standard deviation of 8cm.

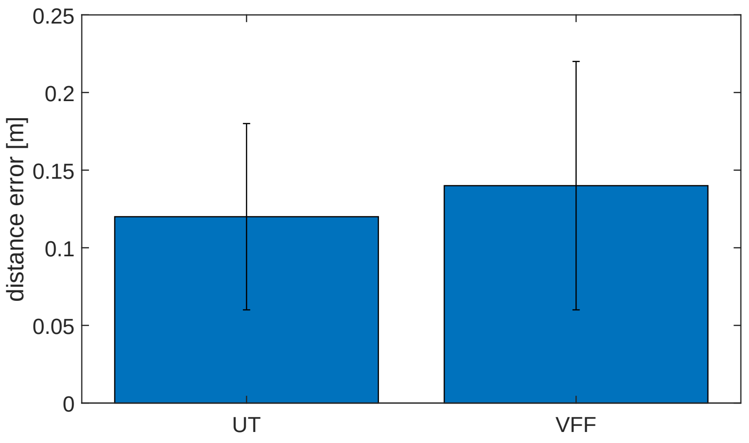

Figure 5.18: This barplot represents the average distance error from the setpoint for the UAV using UT controller and the follower using VFF controller. The error bar indicate the standard deviation value.

5.4. Resume

The VFF controller worked well also in harsh wind conditions. With the leader flying in the straight line, it reached the formation and kept it with a very low steady-state error, the same happens in windy condition, but it just took more time to reach the formation since

the UAVs are flying towards the wind and the maximum speed of the follower is reduced.
Another experiment was establishing the formation while the leader is flying in a circular orbit, here the follower has to follower a continue angle correction, and since the gap makes the orbit of the follower smaller than the orbit of the leader, the roll angle taken by the follower has to be tighter.
At the end we compared the performances of the VFF controller and the UT controller, they got almost the same performance result for some of the tests, but in most of the cases where the follower is far from the leader, the VFF outperformed the UT controller.

6

Conclusion

We started our research project with the idea of trying to apply the vector field approach into a formation fly problem, and explore how this solution can behave in several conditions.
First we defined the behaviours that we expect from a follower to reach the set-point. Then we translated those desired behaviours into control laws by applying the dynamics of the leader and the follower. We analysed the stability of the control solution by applying a mathematical approach and then we also implemented it into a real embedded platform using frameworks that are widely used in industry. moreover, we performed a testing based on the structure software-in-the-loop to validate our code.
At the end, we validated our control solution in physical simulation and we observed that it gives results that are comparable with the state of the art in the research field of formation control, with a light improvement on the convergence time in some conditions if compared to the Unicycle-type Approach control scheme.

7

Future works and improvements

There are many ways to improve the proposed solution with achieve a better performances in a real world application, one of those is the implementation of an adaptive law to estimate the wind action on the UAV and anticipate its correction as shown in [29]. This improvement significantly increase the stability of the system in the situation of strong wind.
Moreover, another improvements of the solution could be to take into account in the equations (3.21) and (3.22) for obstacles, no-fly areas or other agents, such that the system generates a vector field that is also able to avoid obstacles and avoid collisions with the other agents in formation.
The initial plan of this thesis was to test the proposed solution on a real system, but due to some travel restrictions during 2020 was not possible to perform this last crucial part of the thesis.
But this part can be done with a little effort since the system was already tested in SITL and HITL mode.
Another application of this solution, can be used to perform a formation control to land on a moving object, for instance, a ship can be designated as a leader and a UAV be a follower trying to land on this cruising ship.
The landing path and the alignment with the landing track will be maintained by the follower using the vector field for formation control, but a bit of research is needed to control the altitude for the landing and the speed.

References

[1] D. A. Schoenwald, *Auvs: In space, air, water, and on the ground,* IEEE Control Systems Magazine **20**, 15 (2000).

[2] J. Wang and M. Xin, *Integrated optimal formation control of multiple unmanned aerial vehicles,* IEEE Transactions on Control Systems Technology **21**, 1731 (2013).

[3] M. E. Dempsey and S. Rasmussen, *Eyes of the army–us army roadmap for unmanned aircraft systems 2010–2035,* US Army UAS Center of Excellence, Ft. Rucker, Alabma **9** (2010).

[4] A. K. V. de Oliveira, M. Aghaei, U. E. Madukanya, L. Nascimento, and R. Rüther, *Aerial infrared thermography of a utility-scale pv plant after a meteorological tsunami in brazil,* in *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC) (A Joint Conference of 45th IEEE PVSC, 28th PVSEC 34th EU PVSEC)* (2018) pp. 684–689.

[5] K. Swider-Lyons, R. Stroman, M. Schuette, J. Mackrell, G. Page, and J. Rodgers, *Hydrogen fuel cell propulsion for long endurance small uavs,* American Institute of Aeronautics and Astronautics (AIAA) **AIAA Centennial of Naval Aviation Forum** (2011).

[6] P. B. S. Lissaman and C. A. Shollenberger, *Formation flight of birds,* Science **168**, 1003 (1970).

[7] Airbus, *Airlines are looking to reduce fuel consumption. wake-energy retrieval could help,* (2020).

[8] T. J. Mueller, *Aerodynamic measurements at low raynolds numbers for fixed wing micro-air vehicles*, Tech. Rep. (NOTRE DAME UNIV IN DEPT OF AEROSPACE AND MECHANICAL ENGINEERING, 2000).

[9] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, *Vector field path following for miniature air vehicles,* IEEE Transactions on Robotics **23**, 519 (2007).

[10] N. Acosta and J. Toloza, *Techniques to improve the gps precision,* International Journal of Advanced Computer Science and Applications **3** (2012).

[11] M. S. Selig, *Real-time flight simulation of highly maneuverable unmanned aerial vehicles,* Journal of Aircraft **51**, 1705 (2014).

[12] R. Lozano, *Unmanned aerial vehicles: Embedded control* (John Wiley & Sons, 2013).

[13] R. W. Beard, J. Lawton, and F. Y. Hadaegh, *A coordination architecture for spacecraft formation control,* IEEE Transactions on Control Systems Technology **9**, 777 (2001).

[14] T. Balch and R. C. Arkin, *Behavior-based formation control for multirobot teams,* IEEE Transactions on Robotics and Automation **14**, 926 (1998).

[15] B. Issa and A. T. Rashid, *A survey of multi-mobile robots formation control,* International Journal of Computer Applications **181**, 12 (2019).

[16] N. H. M. Li and H. H. T. Liu, *Formation uav flight control using virtual structure and motion synchronization,* in *2008 American Control Conference* (2008) pp. 1782–1787.

[17] H. Mehrjerdi, J. Ghommam, and M. Saad, *Nonlinear coordination control for a group of mobile robots using a virtual structure,* Mechatronics **21**, 1147 (2011).

[18] L. Barnes, M. Fields, and K. Valavanis, *Unmanned ground vehicle swarm formation control using potential fields,* in *2007 Mediterranean Conference on Control Automation* (2007) pp. 1–8.

[19] J. Hu and G. Feng, *Distributed tracking control of leader-follower multi-agent systems under noisy measurement,* Automatica **46**, 1382 (2010).

[20] A. Proud, M. Pachter, and J. D'Azzo, *Close formation flight control,* in *Guidance, Navigation, and Control Conference and Exhibit*.

[21] R. Sattigeri, A. J. Calise, and J. H. Evers, *An adaptive vision-based approach to decentralized formation control,* Journal of Aerospace Computing, Information, and Communication **1**, 502 (2004).

[22] T. Paul, T. R. Krogstad, and J. T. Gravdahl, *Uav formation flight using 3d potential field,* in *2008 16th Mediterranean Conference on Control and Automation* (IEEE, 2008) pp. 1240–1245.

[23] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, *Constrained model predictive control: Stability and optimality,* Automatica **36**, 789 (2000).

[24] S. Farí, X. Wang, S. Roy, and S. Baldi, *Addressing unmodeled path-following dynamics via adaptive vector field: A uav test case,* IEEE Transactions on Aerospace and Electronic Systems **56**, 1613 (2020).

[25] M. R. Rosa, S. Baldi, X. Wang, M. Lv, and W. Yu, *Adaptive hierarchical formation control for uncertain euler–lagrange systems using distributed inverse dynamics,* European Journal of Control **48**, 52 (2019), advanced Control Theory and Applications for Next-Generation Engineered Systems.

[26] K. Sigurd and J. How, *Uav trajectory design using total field collision avoidance,* AIAA Guidance, Navigation, and Control Conference and Exhibit11 August 2003 - 14 August 2003 (2003).

[27] J. Yang, X. Wang, S. Baldi, S. Singh, and S. Farì, *A software-in-the-loop implementation of adaptive formation control for fixed-wing uavs,* IEEE/CAA Journal of Automatica Sinica **6**, 1230 (2019).

[28] D. Kostić, S. Adinandra, J. Caarls, N. van de Wouw, and H. Nijmeijer, *Collision-free tracking control of unicycle mobile robots,* in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference* (2009) pp. 5667–5672.

[29] Bingyu Zhou, H. Satyavada, and S. Baldi, *Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environments,* in *2017 American Control Conference (ACC)* (2017) pp. 1127–1132.

A

Unicycle type approach

In this section we will introduce a tracking control algorithm for a group of unicycle mobile robots as presented in [28]. A centralized system assign to each agent its desired velocity profile and its reference path as a function of the position along the path. The agent motion realize globally asymptotic stable tracking of the reference trajectories under constraints on the actuator inputs. The proposed controller offer a good suppression of the tracking errors due to more freedom in controller tuning and relaxed constraints on the reference velocities.

A.1. Tracking problem

In this solution we aim to design a control law for S and θ such that the robot asymptotically tracks a reference trajectory specified as:

$$P_{ref}(t) = \begin{bmatrix} x_{ref}(t) \\ y_{ref}(t) \\ \theta_{ref}(t) \end{bmatrix} \tag{A.1}$$

Where $x_{ref}(t), y_{ref}(t), \theta_{ref}(t)$ are specified in a world frame. Then we consider tracking errors represented in the UAV frame:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{ref} - x \\ y_{ref} - y \\ \theta_{ref} - \theta \end{bmatrix} \tag{A.2}$$

And the dynamic of the error can be model as:

$$\dot{e}_{xy}(t) = -\omega(t) S e_{xy}(t) + \begin{bmatrix} v_{ref}(t)\cos\theta_e(t) - v(t) \\ v_{ref}(t)\sin\theta_e(t) \end{bmatrix} \tag{A.3}$$

$$\dot{\theta}_e(t) = \omega_{ref}(t) - \omega(t) \tag{A.4}$$

$$e_{xy} = \begin{bmatrix} x_e \\ y_e \end{bmatrix} \quad S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{A.5}$$

A.2. Control law

The following control law solve the given tracking problem:

$$v(t) = v_{ref}(t)\cos(\theta_e(t)) + \phi_{k_x(t),c_x}(c_x x_e(t)) \tag{A.6}$$

$$\omega(t) = \omega_{ref}(t) + \frac{k_y k_{xy} y_e(t) v_{ref}(t)}{\sqrt{1 + (k_{xy} x_e(t))^2 + (k_x y y_e(t))^2}} \frac{\sin(\theta_e(t))}{\theta_e(t)} + \phi_{k_{\theta(t)},c_\theta}(c_\theta \theta_e(t)) \tag{A.7}$$

where k_x, k_y, $k_x y$, k_θ, c_x, $c_\theta \in R_+$ and $\phi_{k_x,c_x} \in S_{k_x,c_x}$, $\phi_{k_0,c_0} \in S_{k_0,c_0}$ are design parameters. We obtain that:

$$\lim_{t \xrightarrow{\infty}} [|x_e(t)| + |\theta_e(t)|] = 0 \tag{A.8}$$

The proof of this statement is clearly explained in [28].