Adaptive Formation Control and Semi-Physical Simulator for Multi-Fixed Wing UAVs

Satish Singh

Delft University of Technology

Delft University of Technology
Master's Thesis in Embedded Systems

Adaptive Formation Control and Semi-Physical Simulator for Multi-Fixed Wing UAVs

Satish Singh
S.Singh-6@student.tudelft.nl

21st November 2019

Thesis by:
Satish SINGH Student Nr. (4705033)

Title:
Adaptive Formation Control and Semi-Physical Simulator for Multi-Fixed Wing UAVs.

MSc presentation:
November 26, 2019.

Graduation Committee:

Dr. ir. Sander WAHLS (Chair)	(DCSC) Delft University of Technology.
Dr. ir. Simone BALDI (Supervisor)	(DCSC) Delft University of Technology .
Dr. ir. Arjan van GENDEREN	(EEMCS) Delft University of Technology.

Acknowledgements

First and foremost, I would like to thank our respected Dr. ir. Simone Baldi, who not only has been my supervisor but also a guardian for the past one year. I can not thank you enough for having me in the team of researchers and pushing me to finish the research paper for the IEEE/CAA Journal of Automatica Sinica and providing me with the opportunity to explore a completely new arena of the scientific world. You have been a true mentor from the very first day and steered me until the finish of this project.
I want to thank especially my parents and my wife who remained glued to me besides all the hardship that came along, and extended extra support whenever needed. My family members Ashok, Sanjay, Shruti, Sangeeta and my special friend Pradeep; I don't even know how to describe how grateful I am for supporting me in all my endeavors and through the darkest part of the night while expecting nothing back from me.
Susanne van Ardene, Mohit, Himanshu, Chinmay, Anup, Georgios, Shivanand, Ninad, Balaji and Philip along with all my friends and colleagues, thank you for supporting me. You have been more than a dear friend all these days. I might fail in doing so, but I shall try to be as good with you as you've always been with me, I shall forever be grateful to you. In your company, I never had a feeling of being in a foreign land. Finally, I want to thank my friends all over the world. Wherever I went, I found many challenges, but also some amazing people. Whenever I faced any tough question there was always someone to answer.

Satish Singh

Delft, The Netherlands
21st November 2019

Abstract

Formation flying is a phenomenon observed very often in the natural world, e.g. birds flying in a flock. The past decade has given a lot of emphasis on the research for the control of autonomous Unmanned Aerial Vehicles (UAVs) of the fixed-wing kind, in an effort to emulate the behavior of natural flocks. Emulating this behavior requires the construction of path following and formation control laws with the capability of adapting to changing situations, in a similar way as natural flocks can do.

This thesis is devoted to studying Adaptive Vector Field Guidance laws and Adaptive Formation laws for fixed-wing UAVs. Formation control relies on an adaptive hierarchical formation control method for uncertain heterogeneous nonlinear agents with Euler-Lagrange (EL) dynamics. It is shown that various formations (T-V-Y formations) can be established using this method, tested using a Matlab/Simulink environment. Additionally, a distinguishing feature of this thesis is the development of a 3D-Simulation platform to perform a hardware in the loop (HITL) simulations (i.e. using the control hardware on board of an actual UAV): a Raspberry Pi is used to run the formation control algorithm and to communicate with a Pixhawk Cube autopilot board which contains the low-level control algorithm. The autopilot board is then connected to a 3D Simulator (Gazebo) and Ground Control System (QGroundControl). The proposed HITL platform promises to facilitate the testing and validation of guidance and formation laws in a much more realistic way than a Matlab/Simulink environment can do.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Family of UAVs can be classified mainly into two categories by their shapes and flying methods: fixed-wing type aircraft and vertical take-off and landing (VTOL) type aircraft. Fixed-wing type aircraft are generally deployed where extensive areas are to be covered in a short time because they can fly with considerable speeds [3]. Nowadays the availability of low-cost sensors, electronics, and air-frames has developed a significant interest in working towards cheaper UAVs among aircraft hobbyists, academic researchers, and industries. UAVs especially fixed-wing kind have been extensively used by military and government organizations in the past. These UAVs were extensively helpful in carrying payloads [4], [5]. Moreover, applications such as mapping, research, and rescue, patrol, etc. require the UAV to autonomously follow a predefined path at a prescribed height [6], [7]. Formation flying of multiple (UAVs) has also become a significant topic of research these days due to numerous defense and commercial applications. To enumerate a few such as reconnaissance and surveillance missions, coordinated attacks, flying into high-risk areas, livestock monitoring, wildfire mapping, etc.

(a) Fixed-wing UAV (b) VTOL Fixed-wing UAV

Figure 1.1: UAV Types *Source: Internet*

1.1 Conceptualizing the problem

Synchronization is the core requirement for the formation flying of the multi-agent system. It is again an emerging field of research which is drawing the attention of not only the control community but many other research fields [8],[9]. The potential solution for coordination among large scale systems can be achieved by network synchronization. Authors of [10],[11] has discussed a solution encompassing spacecraft attitude control for unmanned aerial vehicle. There are two methods to achieve coordination amongst the multi-agent system. Those are either distributed method or a centralized method. On one hand, a distributed approach uses a controller for every agent that utilizes the local information from its neighbors, on the other hand in a centralized approach every node depends on the central node for information hence, the central node dictates all other nodes in the network and control them. A scenario where communication constraint is prevalent the distributed method has an advantage over the central approach [12],[13]. This research work utilizes a distributed synchronization method as a way to control various UAV formations that are accompanied by uncertain parameters (mass, inertia, etc.) with Euler-Lagrange dynamics. Distributed systems have many branches of research area that overlaps networking, control, etc: In distributed control itself while dealing with synchronization many areas can be overlapping, such as, distributed optimization, distributed estimation [14] [15], networked distributed formation also known as flocking with capability in terms of collision avoidance and distributed synchronization, if the synchronization activity is constant then this condition is known as rendezvous or consensus building.

This work deals with one of the aforementioned areas of research: A group of UAVs flying in some formations and traversing a predefined path. It utilizes the distributed formation control law in the inertial frame of reference to accomplish a predefined mission. In this work mainly three types of formation will be discussed those are inverted T (T-formation), inverted V (V-formation) and Y-formation. To attain the formations a formation gap is maintained by the UAVs flocking together. Also, the main feature of this thesis work will be: *the UAVs in the formation can transit from one formation to another while being airborne.*

1.2 State of the art

To discuss a path following strategies, it is necessary to discuss path planning algorithms.

Path planning

A survey on path planning with respect to Autonomous aerial vehicle is discussed in [16], [17] which summarizes state-of-the-art on path following algorithms. *Control effort* and *cross-track error* are the main two metrics on the basis of which authors of [17] has compared the algorithms used in path planing approach. Many path planning algorithms exist in literature to name a few; vector-field (VF), carrot-chasing, nonlinear guidance, pure pursuit with line-of-sight and linear quadratic regulation. While [18],[19] presents adaptive path following strategies, [20] discuss about vector field path following of mini air-vehicles. Authors in [21],[22] has discussed the nonlinear guidance strategy of the UAV. A carrot chasing is the simplest algorithm for path

following and makes the UAV constantly chase a virtual target point and a Line of Sight (LOS) guidance or pure pursuit is a geometric method that chases a virtual target path.

After considering a number of parameter settings, in summary, [17] concludes that Monte Carlo simulations utilizing vector-field algorithm for path-following strategy is much better in terms of accuracy. In comparison with nonlinear guidance strategy (NLGL) and VF path following technique; VF technique developed [20], is more accurate as it has less-error for the same setup in simulations. The only drawback it suffers is the number of design parameters required are higher than other algorithms. The concept behind VF path following is to construct a vector field around the desired path, to command the vehicle in terms of course angle. The stability and convergence to the desired path of the nonlinear vector field path following laws is guaranteed and proved by utilizing Lyapunov stability analysis . Although there have been advances over the years, many issues remain unsolved in the field of path-following. The simulations performed in [17] and in the path-following works points out mainly four key points:

1 The actual performance of path-following methods considerably depends on the fidelity of the UAV model used for design. When aiming at autonomous flight, parametric uncertainties will certainly appear in the UAV structure (uncertain mass and inertia might vary during the mission). Path-following algorithms that cannot adapt to such changes will not sustain and exhibit poor performance [23].

2 The actual path-following performance does not depends only on the commanded course angle. At a lower level, a complex suite of control algorithms commonly referred to as autopilot must be in charge of regulating roll, pitch and altitude (rudder/wing/aileron actuators) according to the course commanded by the path-following algorithm [23].

3 Generally, low-level controllers from autopilot layer [17] is not included into the simulators while testing the path-following algorithms for simulating single UAVs or a flock of UAVs. It makes an important part in hardware in the loop simulations or a real flight: it is also an open research problem to a large extent [23].

4 While investigating control algorithms for UAVs, it is more important to simulate the effect of proposed algorithms in real-time and scenarios as close as to the realistic environment.

For this to achieve, it is required that UAV dynamic behavior of real UAVs in the actual environment is depicted as close as possible in the simulation platform. Moreover, external conditions such as disturbances especially wind must be considered in the controller design.

Unfortunately, quite often a simulation framework that satisfies these requirements is not used by the UAV community. Therefore a mechanism or a simulation platform with the aforementioned properties would be an interesting tool for any researcher involved with UAVs. Often, Matlab and Simulink is used extensively to simulate and visualize along with the testing of control algorithm. Which has certain limitations about involvement of many components. It becomes very cumbersome for Matlab/Simulink in running algorithm, performing simulation, and visualizing the model in real-time. Also it is very difficult to interface the hardware components

to matlab and simulink simulators directly.

Despite the substantial interest in UAVs, little attention has been paid in developing realistic simulators which represents a complete mathematical model of the UAV and the external variables[24], that is the reason why it is important to present a Multi-Aerial Vehicle 3D Simulator that will help with the validation of new controllers.

Simulator

Over many options, to enumerate a few: Unity 3D, X-plane and jMAVSim etc. Gazebo with Robot Operating System (ROS), was given preference to integrate with the this work. Gazebo offers the ability to accurately and efficiently simulate and mimic many types of vehicles and robots including unmanned aerial vehicles in complex indoor and outdoor environments. It has a robust physics engine, high-quality graphics, with convenient programmatic and graphical interfaces. Most importantly, Gazebo is free with a vibrant developer community [25]. Besides being open source, it is a powerful 3D simulation environment for autonomous vehicles and particularly suitable for testing and validation. Simulation is an essential tool in every algorithm developer toolbox. A well-designed simulator makes it possible to rapidly test algorithms, design controllers, and perform regression testing using realistic scenarios. Keeping these features in mind, Gazebo is being used as hardware-in-the-loop (HITL or HIL), software-in-the-loop (SITL) simulator for single vehicle. Gazebo not only supports SITL, HITL and SIH 1.2 simulations for single vehicle but also for multi-vehicle simulation by configuring and interfacing it with ROS.

Simulation types: SITL, HITL and SIH Simulation

Some basic definitions related to simulation types based on their method is given here:

- Software-in-the-loop **(SITL)** simulation, is performed when the flight autopilot stack 3.6.3 runs on computer (either stand alone computer or another computer on the same network). The main advantages of this simulation is that it is faster in comparison to other simulation environments. The fact that all the required component are on the same machine including the ground control stations reduces communication overhead. The disadvantage is of this simulation is it can differ from the actual performance in real scenarios for various reasons. The fact that the performance of a PC or a host machine may not be same as that of the controller used during real flight.

- Hardware-in-the-loop **(HITL)** simulation is performed using a simulation firmware on a real flight controller board attaching it to the simulated environment with the model of the UAV. HITL simulation is a type of real-time simulation used to test the controller design. This simulation shows how the controller responds, in real time for virtual stimulus.
  Advantage of running HITL is that it runs the simulations to show the actual performance of the hardware hence, the flight controller board can directly be integrated to the real UAV after successful simulation. The main issues that can be faced is that baud-rate of sensors and actuators used in simulator can differ to that of real flight. Insufficient Computational

Capability: real-time target machine can lack the computational capability for running the model in real time. If the model fails to run in real time and may produces unreliable results.

- Simulation-In-Hardware **(SIH)** is an alternative to HITL for a vehicle simulation. In this setup, everything runs on embedded hardware - the controller, the state estimator, and the simulator. The machine is only used to display the virtual vehicle, as it is not needed to run the physical plugins as in SITL or HITL. The SIH provides following benefits over the HITL:
  It reduces the communication bandwidth as it ensures synchronous timing by avoiding the bidirectional connection to the computer. Which enables to simulate vehicles without having powerful computers.
  As the complete simulation remains inside the PX4 environment. Hence, it becomes easy to modify the aerodynamic model, or noise level of the sensors, or even add a sensor to be simulated along with incorporating the mathematical model into the simulator.

1.3 Research Objectives

To address these challenges, this thesis work is devoted to answer the following two main research questions: First is related to path following strategy and second is akin to the simulator. For path following algorithm the main objectives are: To deal with parametric uncertainties in the UAV, to cope up with the autopilot low-level control, for testing path-following algorithms and to scale the path-following problem for a group of UAVs. While research work carried by the authors [26], [8] and [27] in adaptive formation control algorithms for various systems with uncertain dynamics, the corresponding problem for UAVs is much more challenging due to the complex UAV control architecture which will be discussed in the following chapters.
In regards to the simulator: the observations prompted to work on a simulation platform with the following features:

- Simulate complete formation model of fixed-wing UAVs;
- Visualize real parameters and models for each fixed-wing UAV components, i.e. control surfaces, motor and propeller model etc.;
- Direct interfacing between real hardware components used onboard such as autopilot and companion computers along with the softwares for simulator and ground control station;

Gazebo is one such open source software that provides such kind of platform which helps in the development of semi-physical simulator for variety of aerial vehicles. This thesis presents the 3D simulator for the multi fixed-wing UAV. It is also multi-aerial because there is a possibility to simulate a fixed-wing as shown in Figure: 1.1(a) or VTOL UAV as shown in Figure: 1.1(b). Along with the aforesaid tasks, this work will also be a contribution towards;

- A software-in-the-loop implementation of adaptive formation control for fixed-wing UAVs based on [23].

- Visualization of the various formations in Matlab/Simulink while the UAVs are airborne.

1.4 Report Synopsis

The rest of the report is organised in chapters as follows:

- ***Chapter 2:*** It starts with the formal presentation of the multiple coordinate frame of references necessary to understand quantities related to an aerial vehicle. Then, Euler-Lagrange dynamics in reference to Fixed-wing UAVs are discussed. Afterwards, the equations pertaining to the forces and moments acting on the vehicle in question are described.

- ***Chapter 3:*** This chapter, firstly discusses the vector field path following strategy for straight line path and orbital path in regards to single UAV. As it is purely mathematical model without any wind dynamics and unmodelled dynamics, hence an adaptive course correction method is introduced to eliminate wind effect along with unmodelled dynamics. Next, the general autopilot control structure used for a fixed-wing UAVs, based on linear controller followed PX4 position and attitude control structure. Afterwards, control algorithm using Inverse Dynamic based method pertaining to Euler Lagrange model of UAVs is discussed in detail. Finally, formation control laws and adaptive synchronization of leader and then follower with reference dynamics is discussed.

- ***Chapter 4:*** This chapter deals with the simulator; starting with the basics of modelling a UAV in Gazebo, to single vehicle simulation environment, to finally multi-vehicle simulation in SITL and HITL environment.

- ***Chapter 5:*** At this stage the simulations are performed for implementation of the algorithms and results are analysed. Hence this chapter comprises of all the simulation results.

- ***Chapter 6:*** A brief conclusion is given here followed by the possibility of works that can be explored in future.

Chapter 2

Modelling Framework for Fixed-wing UAVs

This chapter is organised as: Firstly, a preliminary over orientation of UAV in space is given. In the next section various co-ordinate frame of references required to model a UAV is discussed in brief. In third section, the wind triangle model is discussed followed by a section discussing the mathematical modelling of UAVs in terms of Euler-Lagrange Dynamics. In next three sections communication graphs required for the formation strategies are discussed. Finally, the formation switching topology is also discussed in details.
In this chapter, mathematical modelling of dynamics related to fixed-wing UAV is studied. It enables the development and control the aerial vehicles which are used for controller design. Modelling a dynamical system also enables the possibility to simulate and tune a controller before implementing it on real vehicles.

2.1 Orientation of the UAV

At least two reference frames needs to be considered while working with vehicles moving relative to the earth. The North East Down(NED) frame is commonly chosen for the Earth and BODY for an aerial vehicle. Generally, the NED frame is not considered as inertial frame because of the rotation and velocity of the earth. The NED frame can be simplified and utilized in many tasks and regarded as inertial without introducing too much big of an error. It can be further simplified by looking at the earth as a 2D flat surface, and adding a vertical direction of co-ordinate system as a representation of height coordinate of the NED frame in many cases. The same simplifications can not be done to the body-frame as this frame is accelerated and rotated with respect to the NED frame. Coordinate and velocity transformation matrices is therefore needed. Other reference frames of concern when working with aerial vehicles are wind axis systems, see section: 2.5.
First, let's consider Earth fixed frame $\mathrm{E} = \{E_X, E_Y, E_Z\}$ and body fixed frame $\mathrm{B} = \{B_x, B_y, B_z\}$ on the UAV. At each moment, we need to know the position and the orientation of Body **B** relative to Earth **E**.

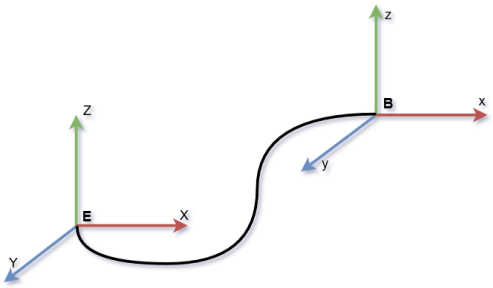


Figure 2.1: General Coordinate System

2.2 Coordinate Frame of References

This section describes the various coordinate systems that are used to describe the position of orientation of aircraft, and the transformation between these coordinate systems. Several different coordinate systems are necessarily used for the following reasons:

- Different onboard sensors take measurements in different reference frames: for example rate gyros take measurements in body frame while GPS takes measurement in inertial frame.
- Equation of motions are applied in the UAVs coordinate frame of reference.
- Mathematical modelling of torques and aerodynamics forces are carried out in body frame.

2.2.1 The Inertial or Earth frame $\mathcal{F}_i$

The inertial coordinate system is an earth-fixed frame with its origin at the defined home location. As shown in the right side of Figure: 2.2, the unit vectors i^i,$j^i$and k^i is directed towards north, east, and the center of Earth, or down. This is commonly used Earth or Inertial frame of reference and referred as the NED (north-east-down) reference frame.

2.2.2 The vehicle frame $\mathcal{F}_v$

The vehicle frame represents the inertial frame translated at the center of mass of the vehicle. The axes of F_v are pointed in the same directions of F_i as shown in Figure: 2.2.

2.2.3 The body frame $\mathcal{F}_b$

The origin of the body frame is also at the center of mass of the UAV. The i-axis points out the nose of the UAV, the j-axis points out the right wing, and the k-axis point out the belly. As the attitude of the UAV moves, the body frame remains fixed with respect to the airframe.

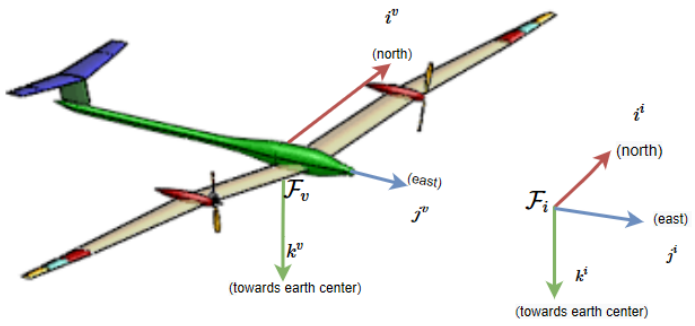


Figure 2.2: Vehicle frame $\mathcal{F}_v$ placed at the UAV's center of gravity (CoG)

Rotation Parametrization and Rotation Matrices: The rotation of a rigid body in space can be parametrized using several methods (Tait-Bryan, Euler angles, quaternion angles etc. "Cardano angles" is a synonym for Tait-Bryan angles that are extensively used in aerospace engineering; and it is normally referred as "Euler angles".
To transform vehicle frame i.e parent frame into the body frame i.e child frame specific rotation matrices are used with angle of rotations namely, ϕ, θ, and ψ about the k, j and i axes respectively. Figure: 2.3 shows a sequential transformation of the frames with their respective rotation angle.

The complete rotation matrix, called Direct Cosine Matrix and is obtained [28] as:

$$\mathcal{R}_v^b(\phi,\theta,\psi) = \mathcal{R}(k,\psi)\cdot\mathcal{R}(j,\theta)\cdot\mathcal{R}(i,\phi) \tag{2.1}$$

$$\mathcal{R}_v^b(\phi,\theta,\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{2.2}$$

$$\mathcal{R}_v^b(\phi,\theta,\psi) = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \sin\phi\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{2.3}$$

The matrix has an unique property:

$$\mathcal{R}_v^b = (\mathcal{R}_b^v)^{-1} = (\mathcal{R}_b^v)^T \tag{2.4}$$

2.2.4 The stability $\mathcal{F}_s$ and the wind frame $\mathcal{F}_w$

The stability and wind coordinate systems is introduced to complete the frame definitions and build a suitable background when treating of aerodynamic forces.

- **Stability Frame**: The velocity of the aircraft with respect to the surrounding air is called airspeed and denoted as $\mathbf{V}_a$. To generate the lift necessary to flight, the wings must fly at

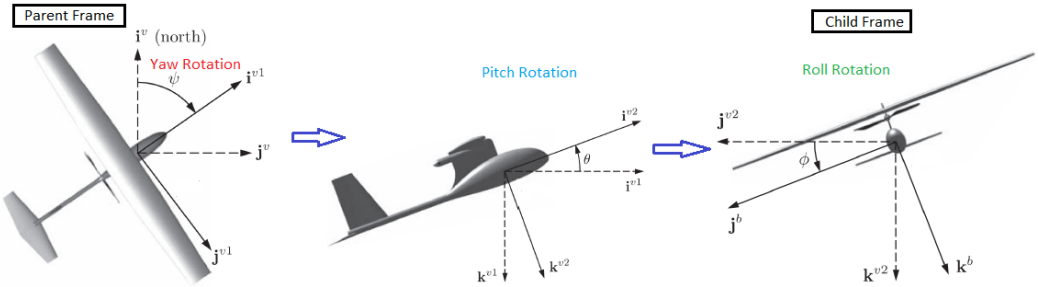


Figure 2.3: Sequential Rotation of Fixed-wing UAV

a positive angle with respect to the airspeed vector. This angle is called angle of attack and denoted with α. It is represented $i^b - j^b$ plane and remains in an angle with airspeed. Hence Stability frame is obtained by rotating the body frame around j^b by the angle of attack α towards the direction of airspeed.

- **Wind Frame**: When airspeed vector does not lie in the $i^b - k^b$ plane, and becomes inclined to the plane the angle of inclination is called as side-slip angle β. By rotating the stability frame by an angle β around k^s axes to get i-axes aligned with the direction of airspeed vector a new frame is obtained. The frame thus obtained is termed as the wind frame.

2.3 The Wind Triangle

Wind has a strong impact on the flight mechanics while studying the aerial vehicle. It may even represent 20%-50% of the airframe airspeed [26]; moreover, the aerodynamic forces depends on the relative speed with respect to surrounding air. Therefore, wind must be properly taken into account while modelling. The wind velocity relative to the inertial frame is hereafter represented by V_w. In the same way, the airspeed with respect to the same frame is denoted with V_a. To account the effects of wind, the ground speed is defined as V_g, hence the UAV velocity relative to the inertial frame is considered. Airspeed, ground speed and wind speed vectors are related by Equation: eq. (2.5). It is briefly accounted for deriving the essential expressions in formulating the equations of motion for an UAV.

$$\mathbf{V}_a = \mathbf{V}_g - \mathbf{V}_w \tag{2.5}$$

Course Angle: To complete the wind triangle represented in Figure: 2.5. The angle between the wind vector $\mathbf{V}_w$ and i^i is denoted by ψ_ω. A new angle χ is introduced, which represents the angle between the true North and the projection of the V_g on the horizontal plane (i^b, j^b). It constitutes the control variable for the the guidance logic.

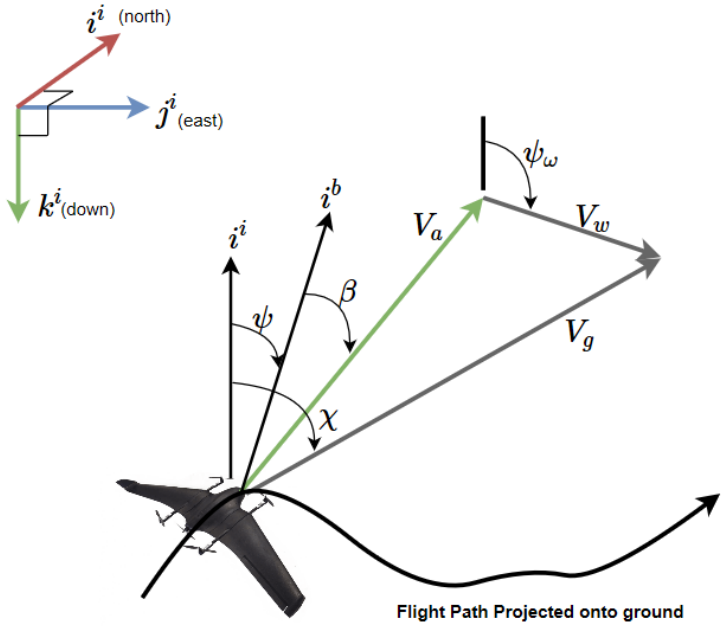


Figure 2.4: The Wind Triangle

2.4 Modelling of UAVs in terms of Euler-Lagrange Dynamics

Let us consider the variables described in Table: 2.1, and axes of motion in body frame represented in Figure: 2.5.

Variable	Description
ϕ	Euler angle for Roll
θ	Euler angle for Pitch
ψ	Euler angle for Yaw
p	Roll rate measured along i^b
q	Pitch rate measured along j^b
r	Yaw rate measured along k^b
u	Body frame velocity along i^b
v	Body frame velocity along j^b
w	Body frame velocity along k^b

Table 2.1: Fixed-wing UAVs parameters

The body-frame angular rates can be expressed in terms of the derivatives of the Euler angles, provided that the proper angular rotational transformations are carried out. A successive rotation around- Roll, Pitch and Yaw axes result in Equation: eq. (2.6), while an inversion of this

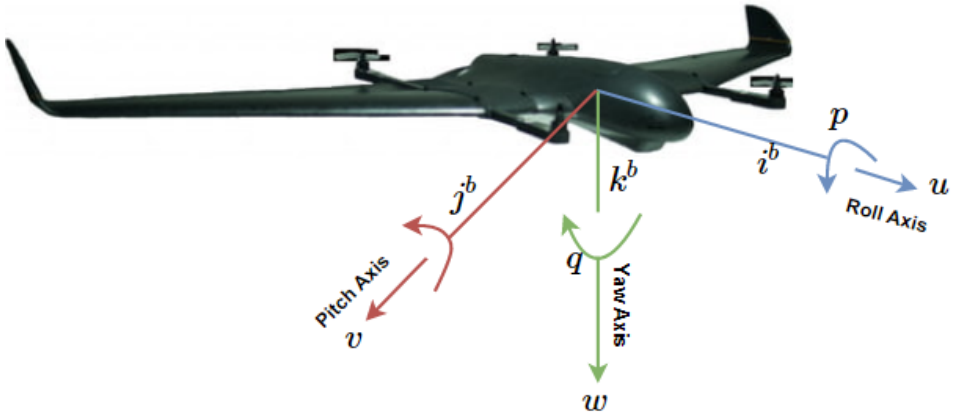


Figure 2.5: Fixed-Wing UAV in Body Frame

Equation yields into eq. (2.7).

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{2.6}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\psi/\cos\theta \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{2.7}$$

Inverted Equation eq. (2.7) reveals that Euler angles representation of the attitude has a mathematical singularity when $\theta = \pm\frac{\pi}{2}$, in which case the yaw angle is not defined.

2.4.1 Equation of motion

The equation of motion are first derived in the inertial (earth) frame, and later the dynamics of inertial velocity will be expressed in body frame.

Translational Motion:
Let, $[\tau_1, \tau_2, \tau_3]^T \in \mathbb{R}^3$ be the sum of forces acting along $[x, y, z]^T$ axis respectively. Then the Euler Lagrange equation for translational motion in the earth frame can be written as:

$$m \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix}^e - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix}^e = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}^e \tag{2.8}$$

where m is the *mass*, the superscript e indicates the earth frame for the UAV, and g is the gravitational constant.The corresponding dynamics of inertial velocity expressed in body frame [29] is:

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & m \end{bmatrix} \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{bmatrix}^b + \begin{bmatrix} 0 & -mr & mq \\ mr & 0 & -mp \\ -mq & mp & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}^b + \begin{bmatrix} mg\sin\theta \\ -mg\sin\phi\cos\theta \\ -mg\cos\phi\cos\theta \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}^b \tag{2.9}$$

where, superscript b expresses quantities in the body frame.

Rotational Motion:
Let, $[\tau_4, \tau_5, \tau_6]^T \in \mathbb{R}^3$ be the sum of moments acting along $[x, y, z]^T$ axis respectively. Then the EL equation for rotational motion in the earth frame can be written as:

$$\underbrace{\begin{bmatrix} I_x & 0 & -I_{xz} \\ 0 & I_y & 0 \\ -I_{xz} & 0 & I_z \end{bmatrix}}_{I} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}^e = \begin{bmatrix} \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}^e \tag{2.10}$$

where I is *inertia tensor*, the fixed-wing UAV is considered to be symmetric with respect to axes **x** and **z**, as well as the fact that the inertia on the planes **xy** and **yz** is negligible hence the I take such a form.
The dynamics of the inertial rotational velocity expressed in body frame [29] is

$$\begin{bmatrix} I_x & 0 & -I_{xz} \\ 0 & I_y & 0 \\ -I_{xz} & 0 & I_z \end{bmatrix} \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}^b + \begin{bmatrix} 0 & I_z r - I_{xz} p & -I_y q \\ -I_z r + I_{xz} p & 0 & I_{xP} - I_{xz} r \\ I_y q & -I_{xP} + I_{xz} r & 0 \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}^b = \begin{bmatrix} \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}^b \tag{2.11}$$

Above Euler Lagrangian Equations: eq. (2.9) and eq. (2.11) can be combined together to be written in a compact form:

$$\underbrace{\begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_x & 0 & -I_{xz} \\ 0 & 0 & 0 & 0 & I_y & 0 \\ 0 & 0 & 0 & -I_{xz} & 0 & I_z \end{bmatrix}}_{D(q)} \underbrace{\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix}}_{\ddot{q}} + \underbrace{\begin{bmatrix} 0 & -mr & mq & 0 & 0 & 0 \\ mr & 0 & -mp & 0 & 0 & 0 \\ -mq & mp & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_z r - I_{xz} p & -I_y q \\ 0 & 0 & 0 & -I_z r + I_{xz} p & 0 & I_x p - I_{xz} r \\ 0 & 0 & 0 & I_y q & -I_x p + I_{xz} r & 0 \end{bmatrix}}_{C(q,\dot{q})}$$

$$\underbrace{\begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix}}_{\dot{q}} + \underbrace{\begin{bmatrix} mgsin\theta \\ -mgsin\phi cos\theta \\ -mgcos\phi cos\theta \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{g(q)} = \underbrace{\begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \\ \tau_4 \\ \tau_5 \\ \tau_6 \end{bmatrix}}_{\tau} \tag{2.12}$$

Where,

Variable	Description
$D(q)$	System inertia matrix(including added mass)
$C(q, \dot{q})$	Coriolis-centripetal matrix (including added mass)
$g(q)$	Vector of gravitational/buoyancy forces and moments
τ	Vector of control inputs
$D\ddot{q}$	Proportional to second derivatives of the generalized coordinates
$C(\dot{q})\dot{q}$	Proportional to first derivatives of the generalized coordinates

Table 2.2: Euler Lagrange Dynamics variable Description

The assumption is that the origin is at the centre of gravity.

2.5 Communication Graphs for the UAVs

communication graph is a network through which the information flow Figure: 2.6 takes place among a flock of UAVs for communicating: the formation gap and/or collision avoidance data. Here, only formation gap data is being used. In such a network, *pinner* node is a special type of node that plays a vital role; also known as reference node. Typically indicated as a system 0; and it can represent a real or a virtual UAV. The main purpose of this node is to generate the data or information for other UAVs in the network. The communication graph describing the allowed information flow between all the systems, pinner excluded, is defined by the pair; nodes and edges given as:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}) \tag{2.13}$$

where, $\mathcal{V} = \{1, ..., N\}$ is a finite non empty set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of pairs of nodes, called edges. To include the presence of the pinner in the network is redefine as:

$$\bar{\mathcal{G}} = \{\mathcal{V}, \mathcal{E}, \mathcal{T}\} \tag{2.14}$$

where, $\mathcal{T} \subseteq \mathcal{V}$ is the set of those nodes, called *target nodes*, which receive information from the Pinner. The next few sections, explains more in details about the communication networks required for various formations.

2.6 Various Formation Strategy (T, V and Y) for a Flock of UAVs

. In this section, the formation strategies for group of UAVs are discussed. For formation, the first things which is necessary among any multi-agent systems is communication on the basis of this fact the chapter starts with the communication graph. Only communication among the multi-agent system is not sufficient to achieve the formation. It must be synchronized to avoid any chaos, hence the next section describes the synchronization of the hierarchical system. Afterwards, the strategies to maintain gaps among the UAVs to achieve the formations is described. In the final section an approach for changing the formations while being airborne is discussed.

2.7 Communication graphs for hierarchical system

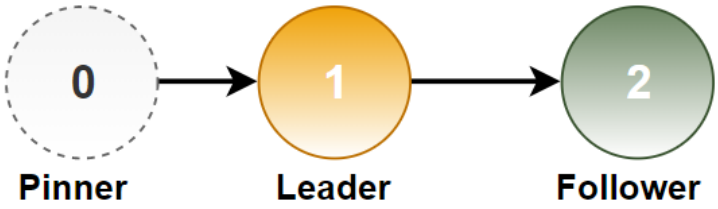


Figure 2.6: UAVs Adaptive Formation communication graph

Let us consider a group of UAVs, tied in a network of UAVs, with $\mathcal{A}$=$[a_{ij}] \in \mathbb{R}^{N\times N}$ be the *Adjacency matrix* of required directed communication graph. Where the elements of $\mathcal{A}$ is defined as $a_{ii} = 0$ and $a_{ij} = 1$, if $(i, j) \in \mathcal{E}$, where $i \neq j$. Also, let us define *target vector* $\mathcal{M} = [a_{j0}] \in \mathbb{R}^N$; to be a vector, the $\mathcal{M}$ establishes the directed communication of for information flow from the Pinner to the target nodes. In this specially defined target matrix: $a_{jo} = 1$ if $j \in \mathcal{T}$ and $a_{jo} = 0$ otherwise.

2.7.1 Communication graph for a flock in T-Formation

Firstly, a communication graph required for T formation strategy is described. It consists of one leader UAV (UAV1) and 3 follower UAVs (UAV2, UAV3 and UAV4) along with the Pinner. For the UAVs to fly and loiter around a given co-ordinate. The Pinner UAV model generates trajectory to which the leader UAV maintain the gap. While followers receive the trajectory

information from leader to maintain the gap with respect to the leader as depicted in the communication graph Figure: 2.7. A configuration needed for T-formation is given in Table: 2.3.

Nomenclature for UAV	Mass (kg)	Moment of Inertia $(kg.m^2)$
UAV-0 (Pinner)	10	$I_x = 0.02$, $I_y = 0.026$ $I_z = 0.053$, $I_{xz} = 0.01$
UAV-1 (Leader 1)	20	$I_x = 0.1$, $I_y = 0.05$ $I_z = 0.1$, $I_{xz} = 0.01$
UAV-2 (Follower 1)	30	$I_x = 0.2$, $I_y = 0.1$ $I_z = 0.2$, $I_{xz} = 0.02$
UAV-3 (Follower 2)	40	$I_x = 0.4$, $I_y = 0.02$ $I_z = 0.4$, $I_{xz} = 0.04$
UAV-4 (Follower 3)	50	$I_x = 0.8$, $I_y = 0.04 I_z = 0.08$, $I_{xz} = 0.08$

Table 2.3: Fixed-wing UAVs parameters for T-formation

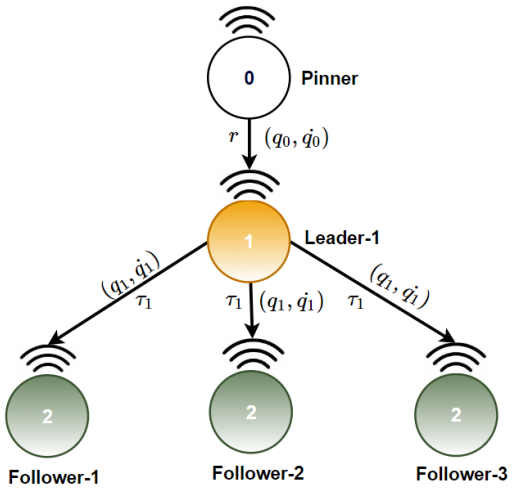


Figure 2.7: Communication Graph for T-Formation

The Adjacency matrix in relation to above network Figure: 1.1(b) can be written as:

$$\mathscr{A}_T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.15}$$

2.7.2 Communication graph for a flock in V-Formation

Here the communication graph required for V formation strategy is described. It consists of two leader UAVs (UAV1 and UAV2) and 2 follower UAVs (UAV3 and UAV4) along with the pinner. Communication graph Figure: 2.8 depicts corresponding to V formation strategy. While Table: 2.4 can be referred for the parameters associated with the UAVs.

Nomenclature for UAV	Mass (kg)	Moment of Inertia $(kg.m^2)$
UAV-0 (Pinner)	10	$I_x = 0.02$, $I_y = 0.026$ $I_z = 0.053$, $I_{xz} = 0.01$
UAV-1 (Leader 1)	20	$I_x = 0.1$, $I_y = 0.05$ $I_z = 0.1$, $I_{xz} = 0.01$
UAV-2 (Leader 2)	30	$I_x = 0.2$, $I_y = 0.1$ $I_z = 0.2$, $I_{xz} = 0.02$
UAV-3 (Follower 1)	40	$I_x = 0.4$, $I_y = 0.02$ $I_z = 0.4$, $I_{xz} = 0.04$
UAV-4 (Follower 2)	50	$I_x = 0.8$, $I_y = 0.04 I_z = 0.08$, $I_{xz} = 0.08$

Table 2.4: Fixed-wing UAVs parameters for V-formation

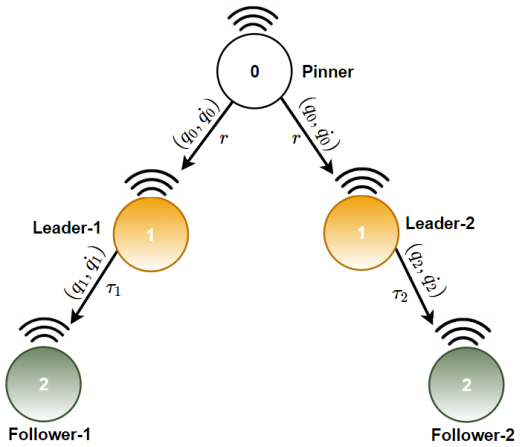


Figure 2.8: Communication Graph for V-Formation

$$\mathscr{A}_V = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.16}$$

2.7.3 Communication graph for a flock in Y-Formation

Finally, the communication graph required for Y formation strategy is given here. With pinner it comprises of three leader UAVs (UAV1, UAV2 and UAV3) and 1 follower UAV (UAV4). Communication graph Figure 2.9 shows the network associated with Y formation. While table 2.5 can be referred for the parameters associated with the UAVs.

Nomenclature for UAV	Mass (kg)	Moment of Inertia $(kg.m^2)$
UAV-0 (Pinner)	10	$I_x = 0.02$, $I_y = 0.026$ $I_z = 0.053$, $I_{xz} = 0.01$
UAV-1 (Leader 1)	20	$I_x = 0.1$, $I_y = 0.05$ $I_z = 0.1$, $I_{xz} = 0.01$
UAV-2 (Leader 2)	30	$I_x = 0.2$, $I_y = 0.1$ $I_z = 0.2$, $I_{xz} = 0.02$
UAV-3 (Follower 1)	40	$I_x = 0.4$, $I_y = 0.02$ $I_z = 0.4$, $I_{xz} = 0.04$
UAV-4 (Follower 2)	50	$I_x = 0.8$, $I_y = 0.04 I_z = 0.08$, $I_{xz} = 0.08$

Table 2.5: Fixed-wing UAVs parameters for Y-formation

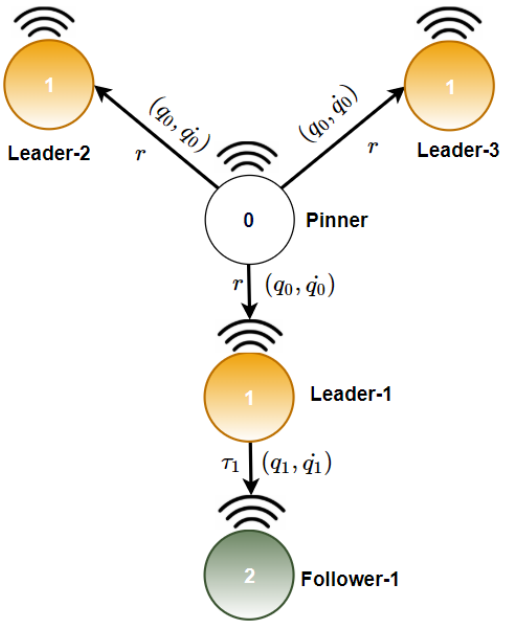


Figure 2.9: Communication Graph for Y-Formation

$$\mathscr{A}_Y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.17}$$

As depicted, all the diagonal elements of the adjacency matrices has only zeros with upper triangular form, this property of a network graph is termed as a hierarchical graph. This graph also contains the root node known as pinner node along with the directed spanning tree. The nodes in the hierarchical networks mutually depends on their neighbours for control inputs as a consequence the nodes in hierarchy exchange control input with the neighbours: the situation of mutual dependence creates an issue of well posedness [30] if the inputs are generated arbitrarily without a prescribed priority; however,[27] shows that with proper modification the distributed model reference adaptive control (MRAC) framework can work, even if the cyclic networks are present.

2.8 Formation Switching Topology

In this section, a very interesting phenomenon is discussed. The fact that the flight controllers can be interfaced with an extra light weight weight computer with increased memory unit. The extra memory and computation power can allow users to have the algorithms related to both type of UAVs (Leader and follower). This can help in switching the role of the same UAV from Leader to follower and vice-versa. Hence it can be fully utilize to alter the hierarchy and the formation as well: that means say UAV starts from ground in T-formation can instantiated from the ground control station to switch the formation to V.
The details of aforementioned idea is as follows: The fact that every UAV has an onboard computer, Raspberry Pi (RPi); and multiple RPi can be configured using a single instance from Ground Control Station (QGC), enables the topology switching during flying condition. To achieve this advance feature, the idea is to use a server/client setup where the UAVs in formation are connected to a server as shown in Figure: 2.10 via wireless access point (AP). All the Raspberry Pis, in formation, are connected to the server program through Wi-Fi. The server program has two functions: Receive user input for specific formation and distribute formation gap data between the UAVs. On receiving these data Raspberry Pi executes the formation control algorithm and issues commands to the autopilot using onboard APIs. Each RPi is configured with static IP and can have both leader and follower dynamics. Thus, hierarchy can be modified while UAVs are airborne. The concept is verified using the Matlab/Simulink Software-in-the-loop simulation. With the requisite preliminary knowledge of various co-ordinate frames, Euler Lagrange Dynamics, and communication graph in relation to Fixed-wing UAVs; in the next chapter algorithms related to path following is discussed.

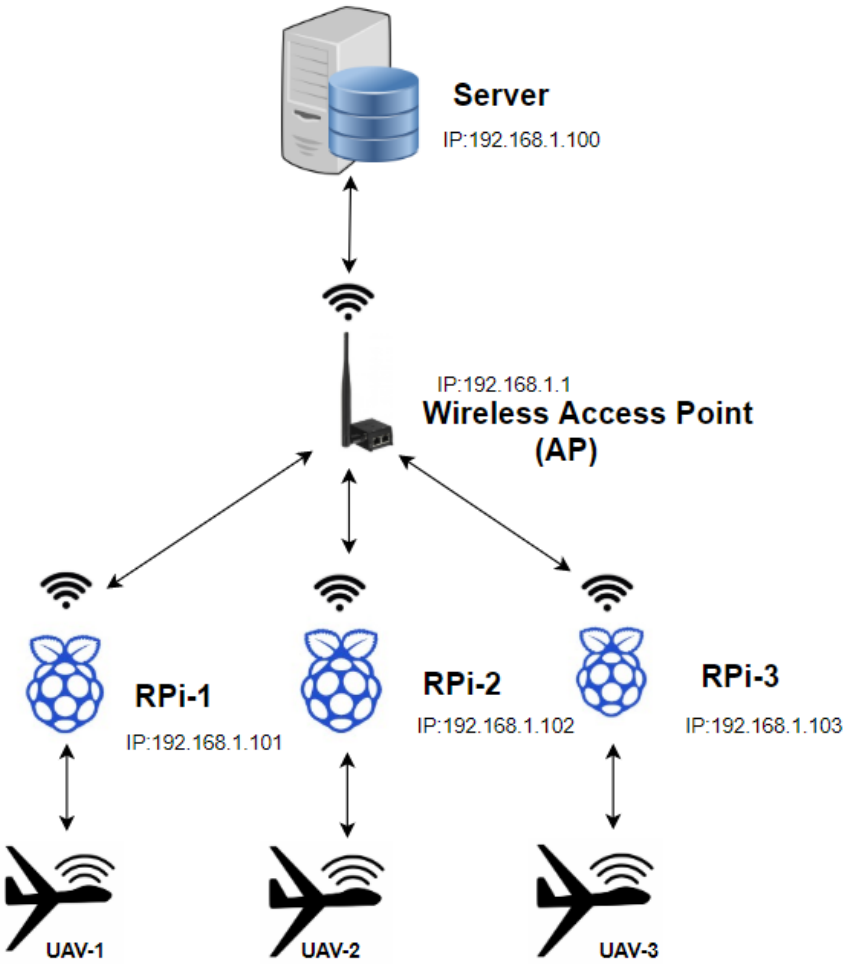


Figure 2.10: Network architecture for topology switching

Chapter 3

Adaptive Vector Field Path Following and Formation Control Algorithm

This chapter is organised as: Firstly, vector field path planning is discussed with respect to primitive path planning methods i.e straight line path and orbital path. Afterwards, adaptive course correction to the vector field algorithm is introduced to compensate for the unmodelled dynamics and wind vectors. In the next section, formation control algorithm for multiple UAVs is brought into prospective. In section four the control law for formation and adaptive synchronization is given in detail. Finally, the PX4 autopilot stack and its low level controllers is discussed.

3.1 Path Following with Vector-Field Approach

In literature a path planning is described as a process of trajectory generation that is traverse by aerial vehicle. Generally, a straight-line and an arc path are considered for path planning [31]. Mission planning can be described as set of waypoint paths predefined for UAV to autonomously traverse and reach the goal.
From the results, it has been shown that Vector Field path following approach achieves the lowest steady-state cross-track error and requires the least control effort with respect to the other approaches [26]. The goal of the Vector Field approach is clearly to drive the cross-track error to zero asymptotically using the course angle χ as the control variable. For this reason, whatever be the UAV's current location with respect to the required path, it is necessary that the the commanded angle χ_c results in the UAV moving towards the path. The angle χ is the most convenient control variable for this objective, since it is present in inertial frame. Considering each point around the desired path, the set of desired course angles is called vector field because it constitutes a set of vectors (relative to the path) of course unit vectors. It is very similar to the artificial potential fields widely used in the robotics community for obstacle avoidance, with the difference in the Vector Field method does not necessarily represent the gradient of a potential; rather, the vector field simply indicates a desired direction of travel. The fundamental paths are the line and the orbit; combining these two primitive path planning approach it is possible to build up more involved paths VF strategies work under the assumption of first-order course χ

dynamics:

$$\dot{\chi} = \alpha_\chi(\chi_c - \chi), \tag{3.1}$$

with χ_c the commanded course angle and α_χ the time constant. In the following sections, the subscript 'sl' will be used to indicate variables referred to the straight-line, while 'o' is used with respect to the orbital path.

3.1.1 Vector Field Strategy for Straight Line Path

The strategy is to construct a vector field such that, when lateral deviation or cross track error say e_{py} is large, the UAV course angle be $\chi_\infty \in (0, \frac{\pi}{2})$ while during the condition when e_{py} approaches zero, the desired course also approaches zero. Let, K_{sl} is a positive constant which regulates the rate of transition from χ_∞ to zero. Large values of K_{sl} result in short, abrupt transitions, while small values of K_{sl} cause long, smooth transitions in the desired course. Hence, the function for the desired course χ_d is:

$$\chi_d\left(e_{py}\right) = -\chi_\infty \frac{2}{\pi} \tan^{-1}\left(k_{sl} e_{py}\right) \tag{3.2}$$

Since, $\chi_\infty \in (0, \frac{\pi}{2})$ the term $\chi_d \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Now, let r is the origin of the path, and $q = [q_n, q_e]$ is the unit direction vector describing the line direction in north and east components. The course angle of the line is given χ_q is $\chi_q = \operatorname{atan} 2\left(\frac{q_e}{q_n}\right)$. If $\chi_q \neq 0$, then Equation: eq. (3.2) is rewritten as:

$$\chi_d\left(e_{py}\right) = \chi_q - \chi_\infty \frac{2}{\pi} \tan^{-1}\left(k_{sl} e_{py}\right) \tag{3.3}$$

It has been proved that when $\chi \to \chi_d$; $e_{py} \to 0$ for $t \to \infty$ [26]. The controller which drives $\chi \to \chi_d$ is described as

$$\chi_c = \chi - \frac{1}{\alpha_\chi}\chi_\infty \frac{2}{\pi} \frac{k_{sl}}{1 + \left(k_{sl} e_{py}\right)^2} V_g \sin\left(\chi - \chi_q\right) - \frac{\kappa_{sl}}{\alpha_\chi} \operatorname{sat}\left(\frac{\tilde{\chi}}{\varepsilon_{sl}}\right) \tag{3.4}$$

where $\tilde{\chi} = \chi - \chi_d$, $V_g = \|\mathbf{V}_g\|$, κ_{sl} and ε_{sl} are two control parameters governing the aggressiveness and counteracting a possible chattering in the control action, and $sat(x) = x$, if $|x| < 1$ or $\operatorname{sign}(x)$ otherwise.The control strategy described above is famously known as sliding mode control [32].

3.1.2 Vector Field Strategy for Orbital Path

The strategy carried out for controller synthesis of course vector field that drives the vehicle to loiter in an orbital path is almost similar to what has been discussed for straight line in the previous section.

$$\chi_d(\tilde{d}) = \gamma + \lambda\left(\frac{\pi}{2} + \tan^{-1}(k_o \tilde{d})\right) \tag{3.5}$$

where, $\tilde{d} = d - \rho$, $\tilde{d}$ is path error. d is the distance of the UAV from the orbit center, ρ is the orbit radius and γ is the angle between the north and the UAV position with respect to the orbit

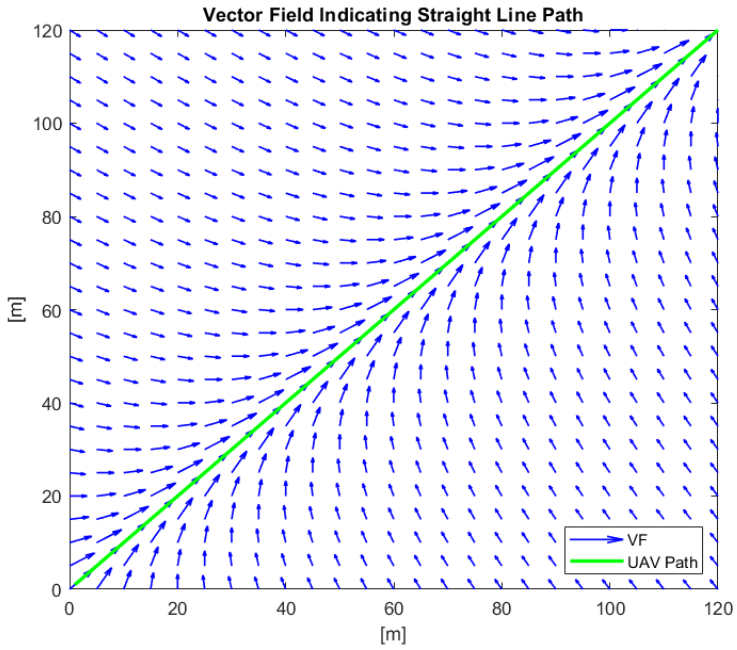


Figure 3.1: VF-Straight Line strategy

center. The parameter λ is 1 for clockwise orbit path. [20] shows that the controller which is able to drive:

$$\chi \to \chi_d \quad and \quad \tilde{d} \to 0 \quad as \quad t \to \infty \tag{3.6}$$

is given as,

$$\chi_c = \chi + \frac{V_g}{\alpha_\chi d}\sin(\chi - \gamma) + \frac{\beta_o \lambda}{\alpha_\chi} V_g \cos(\chi - \gamma) - \frac{\kappa_o}{\alpha_\chi}\,\mathrm{sat}\left(\frac{\tilde{\chi}}{\varepsilon_o}\right) \tag{3.7}$$

where, the parameters k_o, κ_o, ε_o are tuning parameters defined similarly to the straight-line case.

3.2 Adaptive Course Correction for Vector Field Path

The vector field strategy of path following is based on the assumption that the wind vector $V_g = \|\mathbf{V}_g\|$ is known and constant over time. This is an unrealistic assumption which leaves a scope of improvement to both the controllers discussed in Section: 3.1. [18], paves a way to address this short coming in the controller as the known-constant wind hypothesis is relaxed: a new time-varying wind vector is defined and constituted by replacing the constant term. The accuracy can be increased by adaptively tuning some parameters of VF in such a way as to compensate for unmodelled dynamics and unknown wind.

Straight Line Path: The new estimated dynamics for a straight line is given as:

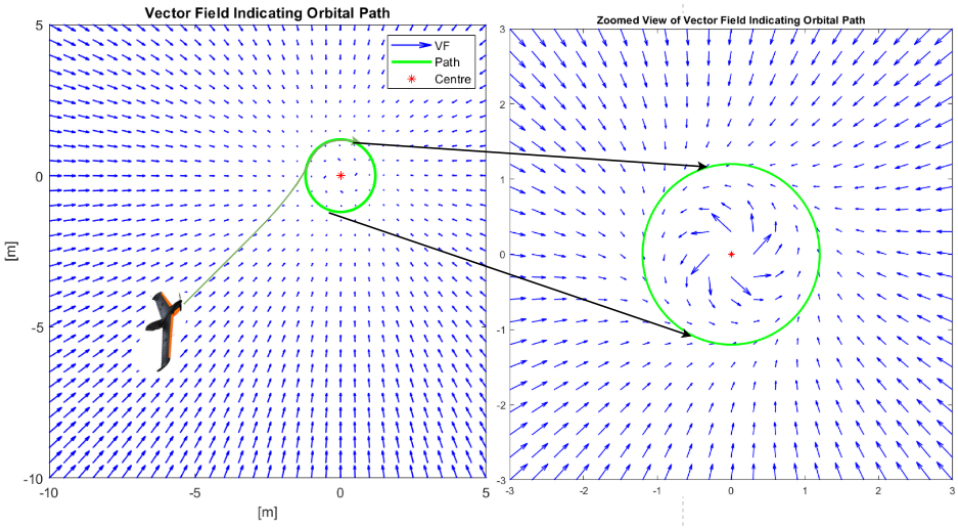


Figure 3.2: VF-Orbital Path strategy

$$\dot{\hat{V}}_g = \Gamma_{sl}\mu_{sl}\tilde{\chi}\chi_\infty\beta_s\frac{2}{\pi}\sin\left(\chi - \chi_q\right) + F_{sl} - \sigma_{sl}\Gamma_{sl}\hat{V}_g \tag{3.8}$$

where, Γ_{sl}, μ_{sl} and σ_{sl} is estimator gain, a weighting term, a damping term respectively, while F_{sl} is a feed-forward term defined as,

$$F_{sl} = \frac{\partial\hat{V}_g}{\partial\chi}\left[-\chi_\infty\frac{2}{\pi}\beta_s\hat{V}_g\sin\left(\chi - \chi_q\right) - \kappa_{sl}\,\mathrm{sat}\left(\frac{\tilde{\chi}}{\epsilon_{sl}}\right)\right] \tag{3.9}$$

Thus,

$$\begin{aligned}\frac{\partial\hat{V}_g}{\partial\chi} \simeq W_s\sin\left(\psi_{w,s} - \chi\right) + \left(V_a^2 - W_s^2\sin^2\left(\psi_{w,s} - \chi\right)\right)^{-\frac{1}{2}} \\ W_s^2\sin\left(\psi_{w,s} - \chi\right)\cos\left(\psi_{w,s} - \chi\right)\end{aligned} \tag{3.10}$$

where, $W_s = ||V_{w,s}||$, $V_{w,s}$ is the velocity of steady-state wind component and $\psi_{w,s}$ is the angle of steady-state wind component with the north direction [33].

Orbital Path: The new estimated dynamics for a orbiting is given as:

$$\dot{\hat{V}}_g = -\Gamma_o\mu_o\tilde{\chi}\left(\frac{1}{d}\sin(\chi - \gamma) + \lambda\beta_o\cos(\chi - \gamma)\right) + F_o - \sigma_o\Gamma_o\hat{V}_g \tag{3.11}$$

Where,

$$F_o = \frac{\partial\hat{V}_g}{\partial\chi}\left[\frac{\hat{V}_g}{d}\sin(\chi - \gamma) + \lambda\beta_o\hat{V}_g\cos(\chi - \gamma) - \kappa_o\,\mathrm{sat}\left(\frac{\tilde{\chi}}{\epsilon_o}\right)\right] \tag{3.12}$$

3.3 Formation for a flock of UAVs

The UAVs form a hierarchical i.e, directed acyclic network while flying namely; *Pinner*, *Leader* and *Follower*. To handle time-varying leader trajectories without sliding mode mechanisms for heterogeneous uncertain UAVs EL dynamics are used. To achieve this, a continuous protocol with the capability of handling large parametric uncertainties and arbitrary leader trajectories considering hierarchical networks is considered. The work in [34] favours the distributed model reference adaptive control (MRAC) with appropriate modifications, also in the presence of cyclic networks. Also synchronization represented here as a solution for coordination of UAVs networked systems. To coordinate multi-agent systems, the distributed approach is used for each agent, a controller that utilizes local information, i.e. neighbours' information. This approach gives more advantages due to its applicability in the presence of communication constraints. In this section, as explained above the formation control used utilizes a graph-based approach. Each UAVs can be represented as a node in the graph. The directed graph edges represents the allowed information flow between the UAV agents (nodes). Figure: 2.6 depicts a graph for a common Formation.

For formation algorithm the UAV Nodes classified into three types are:

- *Pinner Node:* A virtual node, which runs along with the path planning algorithm, having a reference dynamics of an UAV model. Pinner is in synchronism with all other UAVs nodes.
- *Leader Node:* Leaders are the nodes next to the Pinner in the formation. They will receive the data generated by pinner.
- *Follower Nodes:* Followers are the last in the hierarchy which will receive data from neighbouring nodes either from Leader or Companion Follower. Figure: 2.6 shows such a hierarchical representation of UAVs.

3.3.1 Adaptive Formation Algorithm

In this section, algorithm related to heterogeneous UAVs flying in a formation is discussed. A network of UAVs is considered, each one with dynamics:

$$D_i(q_i)\ddot{q}_i + C_i(q_i, \dot{q}_i)\dot{q}_i + g_i(q_i) = \tau_i, \;\; i = \{1, ..., N\} \tag{3.13}$$

where the dynamics are in the EL form as in Equation: eq. (2.12). In state space form, Equation: eq. (2.12) can be rewritten as follows:

$$\underbrace{\begin{bmatrix}\dot{q}\\ \ddot{q}\end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix}0 & \mathbf{1}\\ 0 & -D^{-1}C\end{bmatrix}}_{A}\underbrace{\begin{bmatrix}q\\ \dot{q}\end{bmatrix}}_{x} + \underbrace{\begin{bmatrix}0\\ D^{-1}\end{bmatrix}}_{B}\tau + \begin{bmatrix}0\\ -D^{-1}g\end{bmatrix} \tag{3.14}$$

Following assumptions are made while considering the EL dynamics: The inertia matrix $D_i(q_i)$ is positive definite and symmetric so that the matrix and its inverse are both uniformly bounded. To control the dynamics there exits independent control input for each degree of freedom.

3.3.2 Controller Design for EL Dynamics

To design a controller for the above mentioned dynamical system *Inverse Dynamical Control method is used.* Such controllers are used to mitigate non-linearity of the model present if any, which is given as:

$$\tau_i = D_i\left(q_i\right) a_i + C_i\left(q_i, \dot{q}_i\right) \dot{q}_i + g_i\left(q_i\right), \quad i = \{1, ..., N\} \tag{3.15}$$

Where a_i is defined as,

$$a_i = \ddot{q}^d - K_v \dot{e}_i - K_p e_i \tag{3.16}$$

with an error term $e_i = q_i - q^d$ and proportional and derivative gains k_p and k_v respectively of the multi variable PD controller. The user defined trajectory, velocity and acceleration are represented by variables q^d, $\dot{q}^d$ and $\ddot{q}^d$ respectively. Combination of Equations eq. (3.13) and eq. (3.15) yields:

$$D_i\left(q_i\right)\left(\ddot{q}_i - \ddot{q}^d + K_v \dot{e}_i + K_p e_i\right) = 0 \tag{3.17}$$

where,

$$\ddot{e}_i + K_v \dot{e}_i + K_p e_i = 0 \tag{3.18}$$

The state space form of error dynamics take a form:

$$\begin{bmatrix} \dot{e}_i \\ \ddot{e}_i \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1} \\ -K_p & -K_v \end{bmatrix} \begin{bmatrix} e_i \\ \dot{e}_i \end{bmatrix} \tag{3.19}$$

Hence,

$$\begin{bmatrix} \dot{q}_i \\ \ddot{q}_i \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{1} \\ -K_p & -K_v \end{bmatrix} \begin{bmatrix} q_i \\ \dot{q}_i \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix} \left(\dot{q}^d + K_v \dot{q}^d + K_p q^d\right) \tag{3.20}$$

Equation: eq. (3.20) represents a closed-loop systems of second-order whose state matrix must be Hurwitz by construction. With the knowledge of mathematical modelling and path following for a UAV, now we are ready to discuss various formations in details which is one of the core of this thesis project.

3.4 Formation Control Law and Adaptive Synchronization

Given a hierarchical network $\bar{\mathcal{G}}$ of EL heterogeneous uncertain UAVs, a pinner with state $(q_0, \dot{q_0})$, the task here needs to be addressed is finding a distributed strategy for the inputs τ_i that respects the communication graph, that does not require knowledge of the EL matrices, and that leads to synchronization of the network, in other words; $[q_i, \dot{q_i}] \rightarrow [q_0, \dot{q_0}]$ as t $\rightarrow \infty$ for every UAV i.

3.4.1 Controller for Adaptive Synchronization of Leader to the Reference/Virtual UAV

Here, a control law in relation to leader UAV is framed. To keep things in perspective, leader agent is denoted with index 1. The leader access data from reference or virtual UAV; such as its

trajectory, velocity and acceleration i.e q^d, $\dot{q^d}$ and $\ddot{q^d}$, the main problem leader UAV faces is to cope with uncertainty [8].

$$\begin{bmatrix} \dot{q_0} \\ \ddot{q_0} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \mathbf{1} \\ -K_p & -K_v \end{bmatrix}}_{A_m} \underbrace{\begin{bmatrix} q_0 \\ \dot{q_0} \end{bmatrix}}_{x_m} + \underbrace{\begin{bmatrix} 0 \\ \mathbf{1} \end{bmatrix}}_{B_m} r \tag{3.21}$$

where $q_0, \dot{q_0} \in \mathbb{R}^n$ is the state of the virtual UAV and $r = \ddot{q}^d + K_v\dot{q}^d + K_p q^d$ is a user-specified reference input. The reference dynamics Equation: section 3.4.1 basically represent some homogeneous dynamics to which all the UAVs should synchronize to. In state space form, the leader dynamic can be given as:

$$\begin{bmatrix} \dot{q}_1 \\ \ddot{q}_1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \mathbf{1} \\ 0 & -D_1^{-1}C_1 \end{bmatrix}}_{A_1} \underbrace{\begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix}}_{x_1} + \begin{bmatrix} 0 \\ -D_1^{-1}g_1 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ D_1^{-1} \end{bmatrix}}_{B_1} \tau_1 \tag{3.22}$$

A controller is designed which matches the leader dynamics to the reference model, for this purpose a nonlinear model reference adaptive control method is applied. The ideal controller for this purpose is formulated as:

$$\tau_1^* = \underbrace{\begin{bmatrix} \bar{K}_1^{*\prime} & \bar{K}_1^{*\prime} \end{bmatrix}}_{K_1^{*\prime}} \begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix} + G_1^{*\prime} + L_1^{*\prime} r \tag{3.23}$$

where $*$ indicates an ideal controller for the known system dynamics. The gains that makes the closed loop dynamics of leader matching to the reference model can be calculated by substituting:

$$\begin{aligned} \bar{K}_1^{*\prime} &= -D_1 K_p & L_1^{*\prime} &= D_1 \\ \bar{K}_1^{*\prime} &= -D_1 K_v + C_1 & G_i^{*\prime} &= g_1 \end{aligned} \tag{3.24}$$

Which yield the ideal controller:

$$\tau_1^* = -D_1 K_p q_1 - D_1 K_v \dot{q}_1 + C_1 \dot{q}_1 + g_1 + D_1 r \tag{3.25}$$

The synchronization is achieved adaptively as the system matrices are unknown; and hence for the formation given in Fig. 2.6, the new controller for leader UAV is:

$$\tau_1 = \underbrace{\Theta'_{D1}\xi_{D1}}_{\hat{D}_1}(-K_p q_1 - K_v \dot{q}_1 + r) + \underbrace{\Theta'_{C1}\xi_{C1}\dot{q}_1}_{\hat{C}_1} + \underbrace{\Theta'_{g1}\xi_{g1}}_{\hat{g}_1} \tag{3.26}$$

where, the estimates $\hat{D}_1$, $\hat{C}_1$, $\hat{g}_1$ and $\hat{D}_2$, $\hat{C}_2$, $\hat{g}_2$ of the ideal matrices have been split in a linear-in-the-parameter form.
The adaptive laws for such an estimates are:

$$\begin{aligned} \dot{\Theta}'_{C1} &= -\gamma_1 B'_m \, P \, e_1 \, \dot{q_1}' \xi'_{C1} \\ \dot{\Theta}'_{g1} &= -\gamma_1 B'_m \, P \, e_1 \, \dot{q_1}' \xi'_{g1} \\ \dot{\Theta}'_{D1} &= -\gamma_1 B'_m \, P \, e_1 \, (-K_p q_1 - K_v \dot{q_1} + r)' \xi'_{D1} \end{aligned} \tag{3.27}$$

where θ is the regressand, ξ is the regressor and $e_1 = x_1 - x_m$. Using Lyapunov analysis the estimation laws can be established as, where, γ_1 is adaptive gain and $P = P' > 0$ is such that:

$$PA_m + A_m'P = -Q, Q > 0 \tag{3.28}$$

The linear-in-the-parameter forms for D_1, C_1 and g_1 can be given as:

$$\Theta_{D_1}^{*\prime} = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{x_1} & 0 & -I_{xz_1} \\ 0 & 0 & 0 & 0 & I_{y_1} & 0 \\ 0 & 0 & 0 & -I_{xz_1} & 0 & I_{z_1} \end{bmatrix} \Theta_{g_1}^{*\prime} = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Theta_{C_1}^{*\prime} = \begin{bmatrix} m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{x_1} & 0 & 0 & I_{y_1} & 0 & 0 & I_{z_1} & 0 & 0 & I_{xz_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{x_1} & 0 & 0 & I_{y_1} & 0 & 0 & I_{z_1} & 0 & 0 & I_{xz_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{x_1} & 0 & 0 & I_{y_1} & 0 & 0 & I_{z_1} & 0 & 0 & I_{xz_1} \end{bmatrix}$$

$$\xi_{D_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \xi_{g_1} = \begin{bmatrix} g\sin\theta \\ -g\sin\phi\cos\theta \\ -g\cos\phi\cos\theta \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.29}$$

$$\xi_{C_1}' = \begin{bmatrix} 0 & \bar{r}_1 & -\bar{q}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\bar{r}_1 & 0 & \bar{p}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{q}_1 & -\bar{p}_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{q}_1 & 0 & -\bar{r}_1 & 0 & 0 & \bar{p}_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\bar{p}_1 & 0 & 0 & 0 & \bar{r}_1 & 0 & 0 & -\bar{p}_1 & 0 & \bar{r}_1 \\ 0 & 0 & 0 & 0 & \bar{p}_1 & 0 & -\bar{q}_1 & 0 & 0 & 0 & 0 & 0 & 0 & -\bar{r}_1 & 0 \end{bmatrix}$$

3.4.2 Controller for Adaptive Synchronization of Follower to the Reference Model

Finally the control law that synchronizes follower to reference dynamics via the data accessed from the leader since the reference model signals are not available directly to this UAV, hence the dynamics of the leader UAV act as a reference model. As shown in Figure 2.6 the follower agent is denoted with index 2. The dynamical model for the follower model is:

$$\begin{bmatrix} \dot{q}_2 \\ \ddot{q}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \mathbf{1} \\ 0 & -D_2^{-1}C_2 \end{bmatrix}}_{A_2} \underbrace{\begin{bmatrix} q_2 \\ \dot{q}_2 \end{bmatrix}}_{x_2} + \begin{bmatrix} 0 \\ -D_2^{-1}g_2 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ D_2^{-1} \end{bmatrix}}_{B_2} \tau_2 \tag{3.30}$$

Analogously to the previous section, the aim is to find a matching controller for follower UAVs. The controller given for the follower UAV is:

$$\tau_2^* = \underbrace{\begin{bmatrix} \bar{K}_{21}^{*\prime} & \bar{\bar{K}}_{21}^{*\prime} \end{bmatrix}}_{K_{21}^{*\prime}} \begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix} + \underbrace{\begin{bmatrix} \bar{K}_2^{*\prime} & \bar{\bar{K}}_2^{*\prime} \end{bmatrix}}_{K_2^{*\prime}} \underbrace{\begin{bmatrix} q_2 - q_1 \\ \dot{q}_2 - \dot{q}_1 \end{bmatrix}}_{e_{21}} + G_2^{*\prime} + L_{21}^{*\prime}\tau_1 \tag{3.31}$$

The gains that makes the closed loop dynamics of follower matching to the dynamics of leader can be calculated by substituting:

$$\bar{K}_2^{*\prime} = -D_2 K_p \qquad \bar{K}_{21}^{*\prime} = 0 \quad G_2^{*\prime} = g_2 \tag{3.32}$$

$$\bar{\bar{K}}_2^{*\prime} = -D_2 K_v + C_2 \qquad \bar{\bar{K}}_{21}^{*\prime} = C_2 - D_2 D_1^{-1} C_1 \quad L_{21}^{*\prime} = D_2 D_1^{-1} \tag{3.33}$$

After substituting above gain values in equation 3.31 and the equation henceforth obtained in Equation 3.30 we get Ideal controller for the follower UAV as:

$$\begin{aligned} \tau_2^* &= C_2\dot{q}_1 - D_2 D_1^{-1} C_1 \dot{q}_1 - D_2 K_p \bar{\bar{e}}_{21} - D_2 K_v \bar{e}_{21} + C_2 \bar{\bar{e}}_{21} + g_2 + D_2 D_1^{-1}\tau_1 \\ &= C_2\dot{q}_2 + D_2 D_1^{-1}\tau_1 - D_2 D_1^{-1} C_1 \dot{q}_1 - D_2 \left(K_p \bar{e}_{21} + K_v \bar{\bar{e}}_{21}\right) + g_2 \end{aligned} \tag{3.34}$$

The synchronization is achieved adaptively as again the system matrices are unknown; and hence for the formation given in Figure: 2.6, the new controller for follower UAV is:

$$\begin{aligned} \tau_2 = &- \underbrace{\Theta'_{D_2}\xi_{D_2}}_{\hat{D}_2}\left(K_p \bar{e}_{21} + K_v \bar{\bar{e}}_{21}\right) + \underbrace{\Theta'_{C_2}\xi_{C_2}}_{\hat{C}_2}\dot{q}_2 + \underbrace{\Theta'_{D_2D_1}\xi_{D_2D_1}}_{\widehat{D_2D_1}}\tau_1 \\ &- \underbrace{\Theta'_{D_2D_1C_1}\xi_{D_2D_1C_1}}_{\widehat{D_2D_1C_1}}\dot{q}_1 + \underbrace{\Theta'_{g_2}\xi_{g_2}}_{\hat{g}_2} \end{aligned} \tag{3.35}$$

In short form, Equation: eq. (3.35) can be rewritten as:

$$\begin{aligned} \tau_2 = &- \hat{D}_2[K_p(q_2 - q_1) + K_v(\dot{q}_2 - \dot{q}_1)] + \hat{C}_2\dot{q}_2 \\ &+ \widehat{D_2D_1}_2\tau_1 - \widehat{D_2D_1C}_{12}\dot{q}_1 + \hat{g}_2 \end{aligned} \tag{3.36}$$

where, $\bar{e}_{21} = q_2 - q_1, \bar{\bar{e}}_{21} = \dot{q}_2 - \dot{q}_1$,

$$\begin{aligned} D_2 &= \Theta^{*\prime}_{D_2}\xi_{D_2} \\ C_2 &= \Theta^{*\prime}_{C_2}\xi_{C_2} \\ g_2 &= \Theta^{*\prime}_{g_2}\xi_{g_2} \\ D_2D_1 &= \Theta^{*\prime}_{D_2D_1}\xi_{D_2D_1} \\ D_2D_1C_1 &= \Theta^{*\prime}_{D_2D_1C_1}\xi_{D_2D_1C_1} \end{aligned} \tag{3.37}$$

Where, the estimates $\hat{D}_2$, $\hat{C}_2$, $\hat{g}_2$, $\widehat{D_2D_1}$ and $\widehat{D_2D_1C_1}$ belongs to the ideal matrices $D_2, C_2, g_2, D_2D_1^{-1}$ and $D_2D_1^{-1}C_1$ respectively which have been split in a linear-in-the-parameter form again with, a specific form of regressand Θ and regressor ξ.

The adaptive laws for such an estimates are:

$$\begin{aligned} \dot{\Theta}'_{D_2D_1} &= -\gamma_2 B'_m\, P\, e_{12}\, \tau'_1\, \xi'_{D_2D_1} \\ \dot{\Theta}'_{D_2D_1C_1} &= -\gamma_2 B'_m\, P\, e_{12}\, \dot{q}'_1\, \xi'_{D_2D_1C_1} \\ \dot{\Theta}'_{C_2} &= -\gamma_2 B'_m\, P\, e_{12}\, \dot{q}'_3\, \xi'_{C_2} \\ \dot{\Theta}'_{g_2} &= -\gamma_2 B'_m P\, e_{12}\, \xi'_{g_2} \\ \dot{\Theta}'_{D2} &= -\gamma_2\, B'_m\, P\, e_{12}\, \left[K_p(q_2 - q_1) + K_v(\dot{q}_2 - \dot{q}_1)'\, \xi'_{D_2D_1}\right] \end{aligned} \tag{3.38}$$

where, $\bar{e}_{21} = q_2 - q_1$, $\bar{\bar{e}}_{21} = \dot{q}_2 - \dot{q}_1$, It is proved, the proposed controllers and adaptive laws with all closed-loop signals are bounded, for any Leader UAV:1 and Follower UAV:2, i.e subscript (1,2) such that $e_{12} \neq 0$, there exists $e_{12} = (x_2 - x_1) \rightarrow 0$ as t $\rightarrow \infty$. In addition, for every follower UAV 2 there exists $e_2 = (x_2 - x_0) \rightarrow 0$ as t $\rightarrow \infty$. The proposed synchronization protocol can be extended to include gaps formation, provided that the error:

$$e_{12} = x_2 - x_1 + d_{21} = \begin{bmatrix} q_2 \\ \dot{q}_2 \end{bmatrix} - \begin{bmatrix} q_1 \\ \dot{q}_1 \end{bmatrix} + \begin{bmatrix} \bar{d}_{21} \\ 0 \end{bmatrix} \tag{3.39}$$

is considered, where d_{21} contains the desired formation displacement $\bar{d}_{21}$ among UAVs 2 and 1. The linear-in-the-parameter for the control law is in the form Equation: eq. (3.35), for two UAVs, indicated by subscripts 1 and 2 in Figure: 2.6: it is now shown that the linear-in-the-parameter forms of D_2D_1 and $D_2D_1C_1$ are given as: The linear-in-the-parameter forms for D_1, C_1 and g_1 can be given as:

$$\Theta^{*\prime}_{D_2} = \begin{bmatrix} m_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{x_2} & 0 & -I_{xz_2} \\ 0 & 0 & 0 & 0 & I_{y_2} & 0 \\ 0 & 0 & 0 & -I_{xz_2} & 0 & I_{z_2} \end{bmatrix} \Theta^{*\prime}_{g_1} = \begin{bmatrix} m_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\Theta^{*\prime}_{C_2} = \begin{bmatrix} m_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & m_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{x_2} & 0 & 0 & I_{y2} & 0 & 0 & I_{z_2} & 0 & 0 & I_{xz_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{x_2} & 0 & 0 & I_{y_2} & 0 & 0 & I_{z_2} & 0 & 0 & I_{xz_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{x_2} & 0 & 0 & I_{y_2} & 0 & 0 & I_{z_2} & 0 & 0 & I_{xz_2} \end{bmatrix}$$

$$\xi_{D_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \xi_{g_2} = \begin{bmatrix} g \sin\theta \\ -g \sin\phi \cos\theta \\ -g \cos\phi \cos\theta \\ 0 \\ 0 \\ 0 \end{bmatrix} \xi_{D_2D_1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\xi^{\prime}_{C_2} = \begin{bmatrix} 0 & \bar{r}_2 & -\bar{q}_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\bar{r}_2 & 0 & \bar{p}_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \bar{q}_2 & -\bar{p}_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{q}_2 & 0 & -\bar{r}_2 & 0 & 0 & \bar{p}_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\bar{p}_2 & 0 & 0 & 0 & \bar{r}_2 & 0 & 0 & -\bar{p}_2 & 0 & \bar{r}_2 \\ 0 & 0 & 0 & 0 & \bar{p}_2 & 0 & -\bar{q}_2 & 0 & 0 & 0 & 0 & 0 & 0 & -\bar{r}_2 & 0 \end{bmatrix}$$

$$\Theta^{*\prime}_{D_2D_1} = \begin{bmatrix} \frac{m_2}{m_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{m_2}{m_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{m_2}{m_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{I_{z_1} I_{x_2} I_{xz_2}}{I_{x_1} I_{z_1} - I_{xz_1} I_{xz_1}} & 0 & \frac{I_{xz_1} I_{x_2} I_{xz_2}}{I_{x_1} I_{z_1} - I_{xz_1} I_{xz_1}} \\ 0 & 0 & 0 & 0 & \frac{I_{y_2}}{I_{y_1}} & 0 \\ 0 & 0 & 0 & \frac{I_{xz_1} I_{xz_2} I_{z_2}}{I_{x_1} I_{z_1} - I_{xz_1} I_{xz_1}} & 0 & -\frac{I_{x_1} I_{xz_2} I_{z_2}}{I_{x_1} I_{z_1} - I_{xz_1} I_{xz_1}} \end{bmatrix}$$

$$\Theta^{*}_{D_2D_1C_1} = \begin{bmatrix}
m_2 & 0 & 0 & 0 & 0 & 0 \\
0 & m_2 & 0 & 0 & 0 & 0 \\
0 & 0 & m_2 & 0 & 0 & 0 \\
0 & 0 & 0 & \Gamma_1 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_1 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_1 \\
0 & 0 & 0 & \Gamma_2 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_2 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_2 \\
0 & 0 & 0 & \Gamma_3 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_3 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_3 \\
0 & 0 & 0 & \Gamma_4 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_4 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_4 \\
0 & 0 & 0 & \Gamma_5 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_5 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_5 \\
0 & 0 & 0 & \Gamma_6 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_6 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_6 \\
0 & 0 & 0 & \Gamma_7 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_7 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_7 \\
0 & 0 & 0 & \Gamma_8 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_8 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_8 \\
0 & 0 & 0 & \Gamma_9 & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_9 & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_9 \\
0 & 0 & 0 & \Gamma_{10} & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_{10} & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_{10} \\
0 & 0 & 0 & \Gamma_{11} & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_{11} & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_{11} \\
0 & 0 & 0 & \Gamma_{12} & 0 & 0 \\
0 & 0 & 0 & 0 & \Gamma_{12} & 0 \\
0 & 0 & 0 & 0 & 0 & \Gamma_{12}
\end{bmatrix} \quad \xi'_{D_2D_1C_1} = \begin{bmatrix}
0 & -\bar{r}_1 & \bar{q}_1 & 0 & 0 & 0 \\
\bar{r}_1 & 0 & -\bar{p}_1 & 0 & 0 & 0 \\
-\bar{q}_1 & \bar{p}_1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\bar{q}_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\bar{r}_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bar{p}_1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \bar{q}_1 & 0 & 0 \\
0 & 0 & 0 & 0 & \bar{r}_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\bar{p}_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \bar{r}_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -\bar{p}_1 & 0 \\
0 & 0 & 0 & 0 & 0 & -\bar{q}_1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -\bar{r}_1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \bar{p}_1 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \bar{q}_1
\end{bmatrix}$$

The proposed algorithm can be implemented with a different number of leaders and followers depending upon the computational capability and communication baudrate of the system. Although, the system incorporates the distributed systems, the communication can become a bottleneck in overall performance with higher number of UAVs.

3.5 Autopilot and PX4 controllers

This section presents the general autopilot control structure used for a fixed-wing UAV, based on linear controllers. Getting acquainted with that is necessary to better understand the PX4 controllers structure. The ***autopilot*** is component that is designed to control the attitude and inertial position of an aircraft. To discuss in detail, let us consider a linearised model of the form:

3.5.1 Linearised model

For most flight manoeuvres of interest, autopilots are designed with the assumption of decoupled lateral and longitudinal dynamics. In this way, the structure and the development of an autopilot significantly simplifies and allows the use of successive loop closures, yielding good overall performance. For the lateral dynamics, the variables of interest are the roll angle ϕ, the roll rate p, the yaw angle ψ , and the yaw rate r. The control surfaces used to influence the lateral dynamics are the ailerons and the rudder. Ailerons primarily influence the roll rate p; additionally, both ailerons and rudder influence the yaw angle. Similarly, the variables of interest for the longitudinal dynamics are the pitch angle θ, the pitch rate q, the altitude h and the airspeed V_a. The control

signals used to influence the longitudinal dynamics are the elevator δ_e, and the throttle δ_t. The elevator is used to directly in uence the pitch angle θ. In turn, the pitch angle can be used to manipulate both the altitude and the airspeed. Vice versa, the throttle influences the airspeed and the altitude. Therefore there are some cross talk effects. The linearized lateral and longitudinal dynamics about the equilibrium is given as:

$$\begin{aligned}
\text{Roll angle} \quad & \phi(s) = \frac{a_{\phi_2}}{s\,(s + a_{\phi_1})} \left(\delta_a(s) + \frac{1}{a_{\phi_2}} d_{\phi_2}(s) \right) \\
\text{Pitch angle} \quad & \theta(s) = \frac{a_{\theta_3}}{s^2 + a_{\theta_2} s + a_{\theta_1}} \left(\delta_e(s) + \frac{1}{a_{\theta_3}} d_{\theta_2}(s) \right) \\
\text{Course angle} \quad & \chi(s) = \frac{g}{V_g s} \left(\phi(s) + d_\chi(s) \right)
\end{aligned} \tag{3.40}$$

$$\begin{aligned}
\text{Height (1)} \quad & h(s) = \tfrac{V_a}{s} \left(\theta(s) + \tfrac{1}{V_a} d_h(s) \right) \text{ if } V_a \text{ constant} \\
\text{Height (2)} \quad & h(s) = \tfrac{\theta}{s} \left(V_a(s) + \tfrac{1}{\theta} d_h(s) \right) \quad \text{if } \theta \text{ constant}
\end{aligned} \tag{3.41}$$

where, $d_{\phi 2}(s)$, $d_\chi(s)$, $d_{\theta 2}(s)$, $d_h(s)$ are disturbances coming from the coupled dynamics and wind action. Coefficients $a_{\phi 1}$, $a_{\phi 2}$, $a_{\theta 1}$, $a_{\theta 2}$ and $a_{\theta 3}$ are scalars from the linearization. Such first or second order loops allow an effective use of Proportional-Integral-Derivative (PID) control.

Because the purpose of the autopilot layer is to provide low-level controllers to govern the various UAV states, let us now consider the PX4 control scheme.

3.6 PX4 Controller Scheme

This section comprises of control loop diagrams for main PX4 controllers as implemented in autopilot stack. Autopilot is incorporated in flight modes and responds in closed loop or to user input and controls the vehicle movement. Flight modes are loosely grouped into with or without human assisted: and basically named as; assisted, manual, and auto modes, based on the level/type of control provided by the autopilot. The autopilot transitions can be instantiated between flight modes using ground control station. Controllers used for the vehicle movement of fixed-wing UAVs are discussed here in detail.

3.6.1 PX4 Fixed-wing position controller

In this section, implementation of the position controller in PX4 is discussed. It utilizes total energy control system commonly known as (TECS). It is implemented as a library which is used in the fixed-wing position control module. It enables simultaneous control of true airspeed and altitude of a fixed-wing aircraft.

Figure: 3.3 shows architecture related to TECS. As shown here, altitude and airspeed setpoints are inputs while pitch angle and throttle setpoints are outputs from the control systems. Both outputs are are used as input of the fixed wing attitude controller. The performance of th pitch control loop directly affects the performance of TECS. Hence, a a poor tracking of the aircraft pitch angle [1] often times causes a poor tracking of airspeed and altitude.

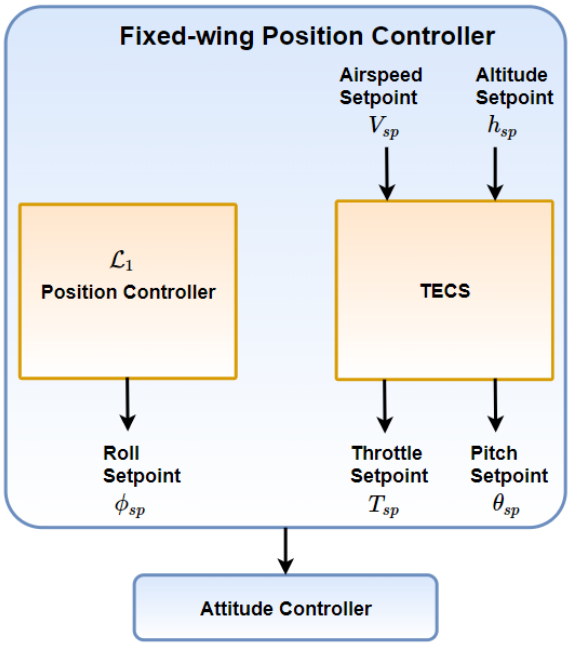
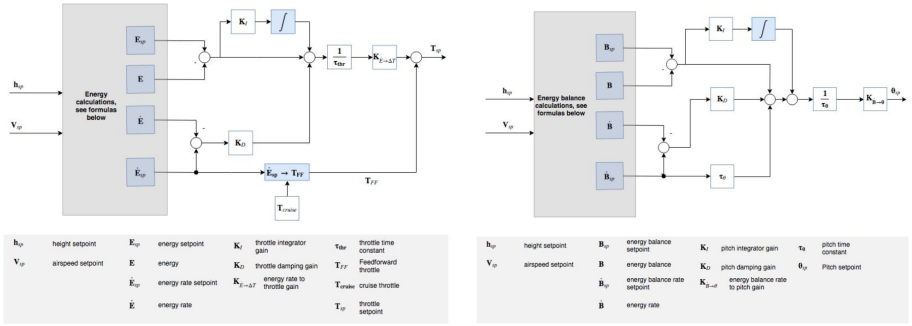


Figure 3.3: PX4 Fixed-wing general position control scheme



(a) Total energy control loop of TECS controller

(b) Total energy balance control loop of TECS controller

Figure 3.4: TECS controllers *source: [1]*

The control of height h and true airspeed V_a simultaneously is non-trivial in a sense that an increase in the pitch angle θ_{sp} causes an increase in height at the same time it decreases the airspeed of the UAV while an increase in the throttle increases the airspeed and height due to increase in lift. Therefore, it has two inputs (pitch angle and throttle) both of which affects the two outputs airspeed V_a and altitude h this coupled dynamics makes the control challenging. To solve this coupled dynamics: A regime division can be implemented: climb zone, altitude hold zone, descent zone. A full throttle in the climb zone and zero throttle in the descent zone can be used to regulate the airspeed with the pitch angle. In the altitude hold zone, the simplest and most

effective method is to regulate altitude by commanding pitch, and the airspeed by commanding the throttle.
General control scheme Figure: 3.3, shows such an architecture where, airspeed dictates of throttle setpoint while height dictates the pitch setpoint. However, from Equation: eq. (3.41) it is evident that the altitude dynamics and the airspeed dynamics are coupled. Which means the aircraft's kinetic energy is converted to potential energy as; airspeed decreases with increase in altitude and vice-versa. The decoupled dynamics lead to a poor performance with respect to other methods based on energy considerations. One of the deeply analysed and improved method is Total Energy Control Systems, or TECS [35]. In the following, the Pixhawk/PX4 implementation of TECS is described [1].

By definition, the total energy of an UAV is the sum of kinetic energy and potential energy:

$$E_T = \frac{1}{2} m V_a^2 + mgh \tag{3.42}$$

The total energy rate of the UAV is given by taking the derivative with respect to time:

$$\dot{E}_T = m V_a \dot{V}_a + mg\dot{h} \tag{3.43}$$

Hence, the specific energy rate can be given as:

$$\dot{E} = \frac{\dot{E}_T}{mgV_a} = \frac{\dot{V}_a}{g} + \frac{\dot{h}}{V_a} = \frac{\dot{V}_a}{g} + \sin(\gamma) \tag{3.44}$$

where, γ is the flight plan angle. For small γ an approximate specific energy rate can be given as:

$$\dot{E} \approx \frac{\dot{V}_a}{g} + \gamma \tag{3.45}$$

From the dynamic equations of an aircraft we get the following relation:

$$T - D = mg\left(\frac{\dot{V}_a}{g} + \sin(\gamma)\right) \approx mg\left(\frac{\dot{V}_a}{g} + \gamma\right) \tag{3.46}$$

where, T and D are the thrust and drag forces. In level flight, initial thrust is trimmed against the drag and a change in thrust results thus in:

$$\Delta T = mg\left(\frac{\dot{V}_a}{g} + \gamma\right) \tag{3.47}$$

As it can be seen, ΔT is proportional to $\dot{E}$, hence altering the thrust will proportionally alter the specific rate of energy into the UAV, which will increase the sum of the flight path angle and the acceleration along the flight path. Therefore the thrust setpoint should be used by the total energy control system. Elevator control on the other hand is energy conservative, and is thus used for exchanging kinetic energy for potential energy and vice-versa. Hence, an specific energy balance rate can be defined as:

$$\dot{B} = \gamma - \frac{\dot{V}_a}{g} \tag{3.48}$$

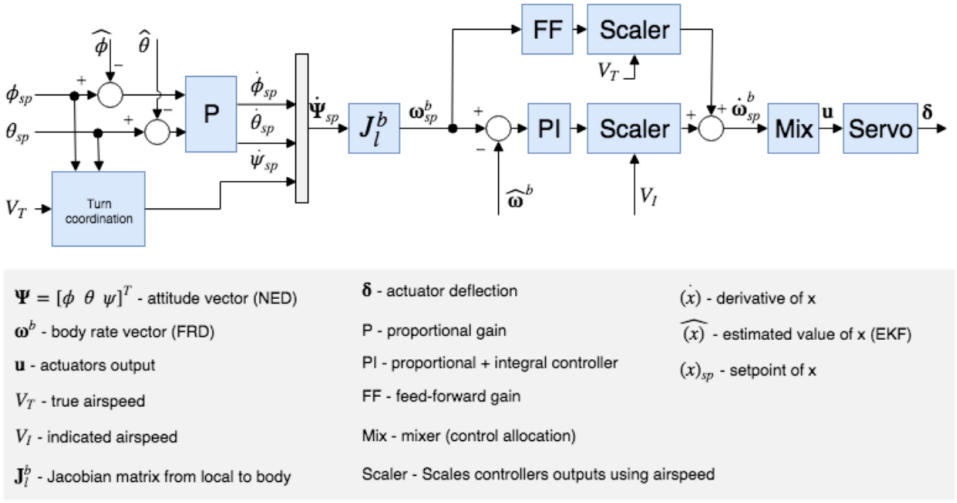


Figure 3.5: Fixed-wing attitude controller *source: [1]*

3.6.2 PX4 Fixed-wing attitude controller

In PX4 autopilot stack, discrete time control loops for pitch, roll and yaw is implemented: A discrete time cascaded control loop for attitude control is realised as shown in Figure: 3.5. In the outer loop of this controller, a constant gain is multiplied with the calculated error between the attitude setpoint and the estimated attitude. In other terms; a P controller is used here to generate a rate setpoint. In inner loop of the controller, the error in rates is calculated, which is then used with a Proportional and integral (PI) controller to generate desired angular acceleration [1].

Utilizing the desired angular acceleration and a priori knowledge of the system, the angular position of the control effectors of the fixed-wing UAV (rudder, ailerons, elevators or elevons etc.) is then computed through control allocation which is also known as (**mixing**). The gain scheduling for the Fixed-wing attitude controller is disabled in the absence of airspeed sensor therefore it works as an open loop system and aerodynamic damping can be compensated using feedforward gain. Basically, the two main components of inertial frame moments on an aircraft are produced by the control surfaces (producing the motion - elevators, rudders, ailerons) and the aerodynamic damping (counteracting the motion - proportional to the body rates). In order to keep a constant rate, this damping can be compensated using feedforward in the rate loop. The longitudinal and lateral dynamics are assumed to be decoupled enough to work independently and the roll and pitch controllers have the same structure Figure: 3.6 . The yaw controller, generates yaw rate setpoint using the turn coordination constraint in order to minimize lateral acceleration, generated when the aircraft is slipping. It also helps to counteract adverse yaw effects and to damp the Dutch roll mode by providing extra directional damping [1].

3.6.3 PX4 Autopilot Flight Stack Software: an overview

The flight stack is a collection of guidance, navigation and control algorithms for autonomous vehicles which processes the information from input to output inside the control board. It

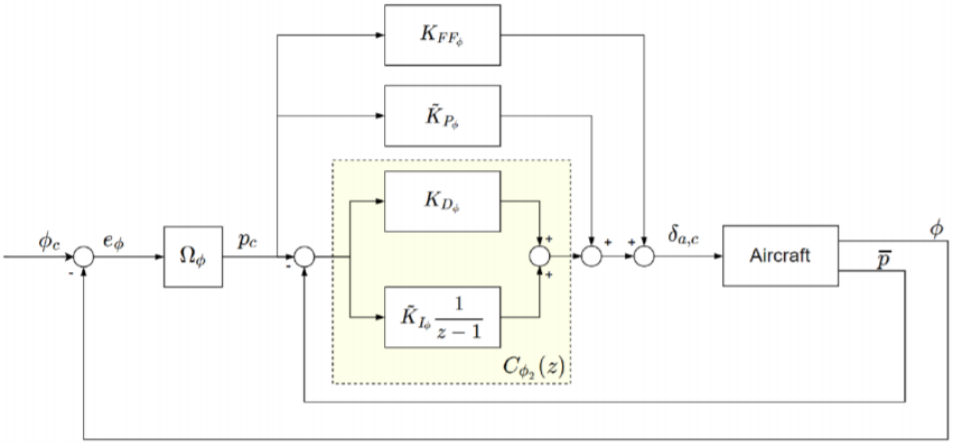


Figure 3.6: The controller scheme for roll control of Fixed-wing UAV. The variables ϕ, ϕ_c, $\bar{p}$, p_c and $\delta_{a,c}$ are the roll angle, commanded roll angle, roll rate and commanded roll rate and commanded aileron angle respectively

includes controllers for fixed wing, multirotor and VTOL airframes as well as estimators for attitude and position. The following diagram shows an overview of the building blocks of the flight stack. It contains the full pipeline from sensors and autonomous flight control (Navigator), down to the motor or servo control (Actuators).

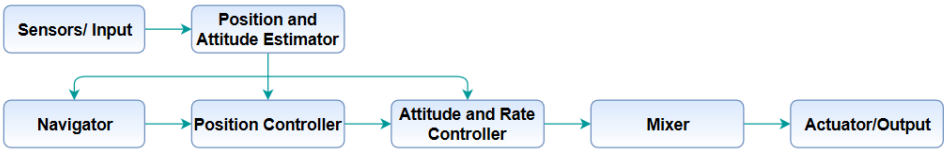


Figure 3.7: PX4 autopilot flight Stack for autonomous flying of a single vehicle

An ***estimator*** is a component that takes one or more sensor inputs, combines them, and computes a vehicle state. While ***controller*** is a component that takes a set-point and a measurement or estimated state or a process variable as an input. The main goal of a controller is to adjust the value of the process variable such that it matches the set-point. The output is a correction to eventually reach that set-point. A ***mixer*** takes force commands and translates them into individual motor commands, while ensuring that limits are not exceeded.
Next is various simulation environment for VTOL vehicle. To achieve this, setting up a Ubuntu machine accompanying all the required softwares is first setup towards the process. For this project Ubuntu LTS 18.04 machine is used as a host along with ROS melodic accompanying Gazebo 9.0 and PX4 flight stack.

Chapter 4

Gazebo: As semi-physical simulator

This chapter discusses the simulation platform that can be used for simulating and visualizing the multiple UAVs and it is organised as: Beginning with simulator description format for a robot modelling in this case an UAV, later on in the next section, the discussion about simulation environment for single vehicle followed by multi-vehicle simulation environment is carried out. Lastly, the a simulator that has been tested with configuration containing companion computer is discussed.//

4.1 Modeling a UAV in Simulation Description Format

Gazebo uses, Simulation Description Format **(SDF)** for modelling a vehicle or robot: it is an XML format that describes objects and environments for UAV simulators, visualization, and control [36]. It was developed as part of the Gazebo robot simulator, SDF is designed for the use of various scientific simulation applications. Due to continuous development in past few years, SDF has become a stable, robust, and extensible format capable of describing every aspects from robotic models, static and dynamic objects, to lighting, terrain, and even physics. A short premise on different elements and its description is given below:
Figure 4.1 shows root elements of sdf and its description which are basic building blocks for modelling a robot, in this case an UAV. The root contains a tree of elements for example: ***world*** and world element encapsulates an entire world description including: models, scene, physics, joints and plugins. A ***model*** element defines a complete UAV or any other physical object. A model may contain many ***links*** and ***joints***: Link is a physical link with inertia, collision, and visual properties while a joint is a joint connection of two links with kinematic and dynamic properties. A link and joint may contain many ***sensors***, the sensor tags describes the type and properties of a sensor. A link may also contain many ***collisions*** and ***visuals***; Collision shows collision properties of a link this can be different from visual properties of a link, a simple collision models are often used to reduce computation time. Visual is a properties of link that specifies the shape of the objects (box, cylinder, etc.) for the purpose of visualization of a model. A visual may contain one ***material*** which describes the material of visual element. Finally, the visual and collision is accompanied with ***geometry*** as a child element. Next is ***Scene***, it specifies the look of the environment such as clouds, speed and density of clouds. A world can have only

one ***physics*** element which specifies the type and properties of dynamics engine.A world may contain many ***light*** elements inside a scene. An sdf file can contain only one ***actor*** but a world can have many actors, it is a special kind of model which can have a scripted motion. This includes both global waypoint type animations and skeleton animations.

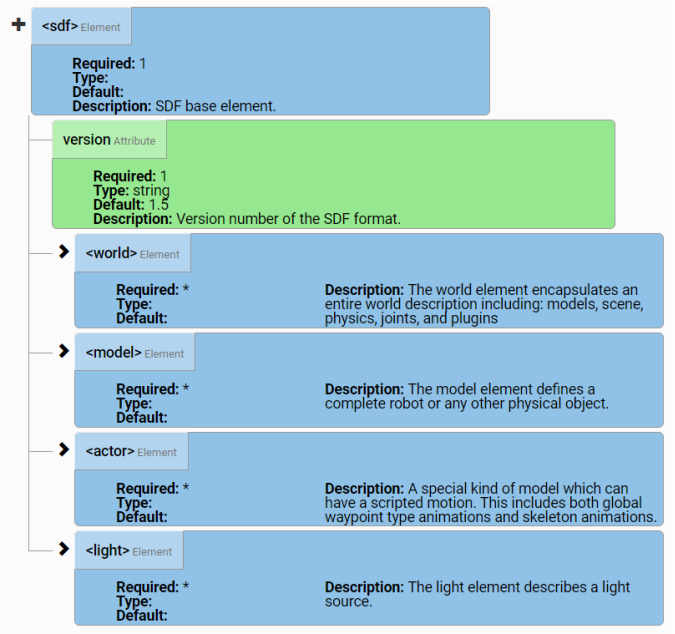


Figure 4.1: SDF Architechture *source:* [2]

In the next few sections, the simulator is discussed only with respect to fixed wing UAVs specially VTOL type.

4.2 Single UAV Simulator

4.2.1 SITL Environment for Single Vehicle

For software-in-the-loop simulation of a single fixed-wing UAV the softwares used are; Gazebo as simulator, PX4-(v2)/cube (black) firmware as an autopilot stack and QGroundControl as a ground control station (GCS). The brief procedure is mentioned Appendix: A. It is necessary to point out that ROS is not mandatory for single vehicle SITL or HITL simulations. An interfacing architecture for such configuration is shown in Figure: 4.2. The different parts of the system such as simulator, QGC and is connected via UDP (User Datagram Protocol) communication protocol, PX4 uses a simulation-specific module to listen to TCP (Transmission Control Protocol) and UDP commonly known as MAVLink.
MAVLink or Micro Aerial Vehicle Link is a communication protocol [37]. It is specially developed for communicating with aerial robots or drones and its components (onboard autopilot and ground control station). It is a very light weight messaging protocol. It follows a modern

hybrid publish-subscribe and point-to-point design pattern: Data streams are sent / published as topics while configuration sub-protocols such as the mission protocol or parameter protocol which are point-to-point with retransmission.
To define the messages in XML files are used. According to particular mavlink communication protocol the messages are defined in each XML files, also termed as **"dialect"**. PX4 firmware mavlink communication module contains a file named common.xml, where set of reference messages that needs to be implemented in autopilots and ground control system are defined. Many dialect are build according to these definitions.

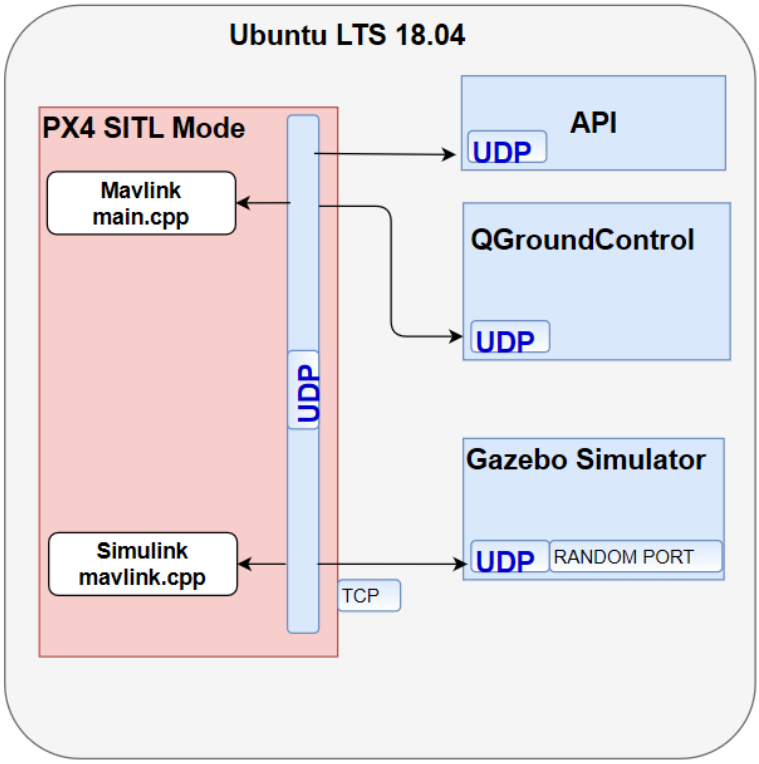


Figure 4.2: SITL Architecture

The next step is to build and launch the vehicle; for all purposes of this project VTOL(vertical take off and landing) model, which combines the feature of both quad-rotor and fixed-wing UAV is used. Firstly, QGroundControl was configured for SITL afterwards standard vtol model was build and launched as explained in APPENDIX: A. Afterwards, the path was planned using waypoints such as straight line path and loiter points as shown in Figure: 4.3. Finally, the model was given a mission to fly to as shown in Figure: 4.4.

(a) Straight line path mission (b) Straight line path with loitering mission

Figure 4.3: Mission for single vehicle SITL for VTOL

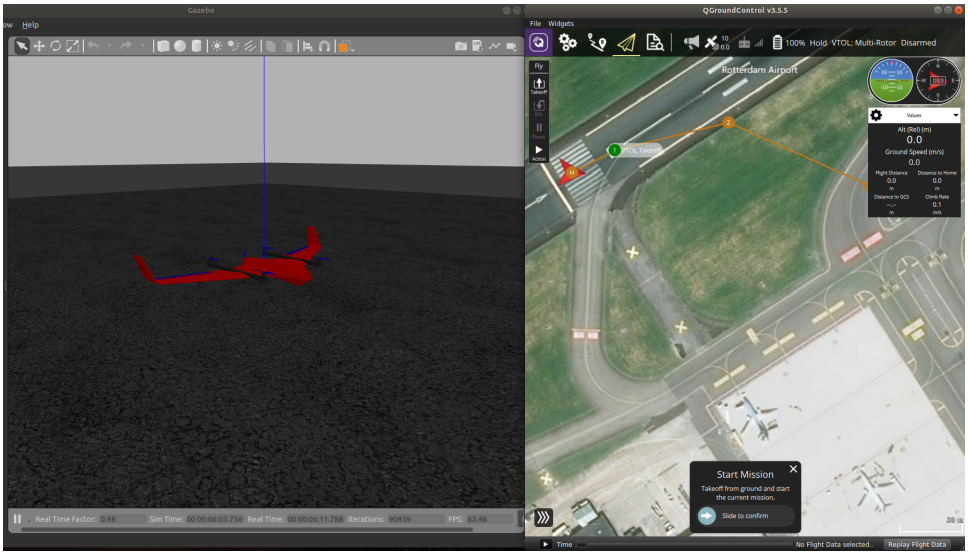
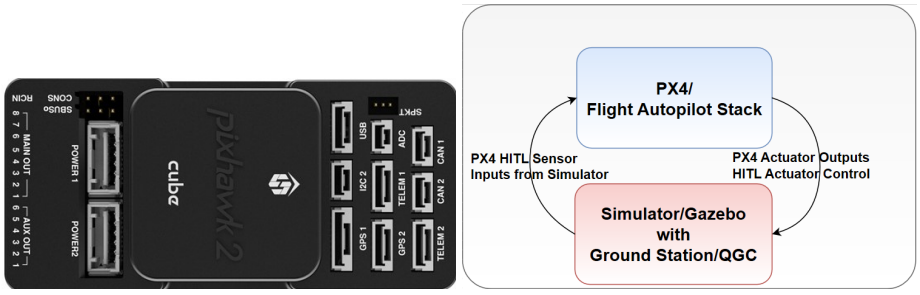


Figure 4.4: Simulator (Gazebo) with ground control station (QGC)

4.2.2 HITL environment for Single Vehicle

HITL for a single fixed wing UAV requires all softwares used in the SITL along with the hardware, PX4 Cube Figure: 4.5(a). There are three ways HITL simulation are performed; in this report only two methods will be mainly discussed. One way to achieve HITL is to attach the Pixhawk board through USB cable to Ubuntu machine and configure the simulator as shown in architecture Figure: 4.5(b). In this configuration the autopilot stack is build and uploaded onto the hardware itself opposite to that of SITL; where autopilot stack also runs on the host machine. The simulator runs the physical model of the UAV and sensor outputs are conveyed to controllers via USB while PX4 board acknowledge the inputs and generates the actuator output for simulator completing the loop. The PX4 and QGC and Simulator configuration setup is explained in [38].
Another way to achieve HITL is to use a companion computer along with PX4 board. This method will be explained in the last section of this report.



(a) PX4/Pixhawk Cube (b) HITL Configuration

Figure 4.5: Single vehicle HITL for VTOL

4.3 Multi-UAV Simulator

This section explains the method of simulation of multiple UAVs using Gazebo 9.0 and ROS Melodic. An example setup that opens the Gazebo client graphical user interface (GUI) showing five standard vtol vehicle model Figure: 4.6 in an empty world is described here.

4.3.1 SITL environment for Multi-UAV

This section explains, the work devoted towards developing a multi- UAV SITL environment. To achieve the goal, MAVROS [39] package is required along with the other software packages discussed in previous section.
MAVROS is a package that utilizes mavlink communication protocol [40] to provide communication driver for various autopilots. Additionally, it also bridges the link between UDP and mavlink for ground control stations. UAVs are then controlled with QGroundControl and MAVROS in a similar way; as a single vehicle is manage for SITL single vehicle simulation which has been already explained in section: 4.2.1 . For simulation of each vehicle, the following is required:

Gazebo model: This is defined in simulation description format (SDF) file in Firmware folder of PX4 module. This SDF file is used to generate Gazebo visualization model (UAV) and also contains the communication protocol definition for every vehicle model to be selected during simulation. A mavlink protocol defined with argument (mavlink_udp_port) inside the SDF file of Gazebo model bridges the communication with PX4. For a multi-vehicle simulation the UDP port is defined in the main launch file. If the model being launched is of same type, separate launch file is not needed for the number of vehicles. The same SDF file is adequate to generate five vehicles.

PX4 node: This is the PX4 SITL application. The UDP port is defined and set in the start up file in the application side (SITL_UDP_PRT) parameter in a similar fashion as that of simulator. As explained in the previous section mavlink protocol bridges the same UDP port defined

(mavlink_udp_port) in the simulator end to communicate with the autopilot; Gazebo with ROS and/or Gazebo (Simulation vehicle model). Both files must be set to have the same port configuration as discussed previously for the mavlink_udp_port. On the basis of vehicle type argument, the path in the start-up file is generated. For mutiple vehicles the launch file must also contain the same vehicle type and vehicle ID. To make sure the configurations consistency, while describing the vehicle ID, every vehicle to be launched with the MAV_SYS_ID in the start-up file must match with the vehicle ID inside launch file. Figure 4.7 shows the the ground control station and gazebo with the UAVs during flight.

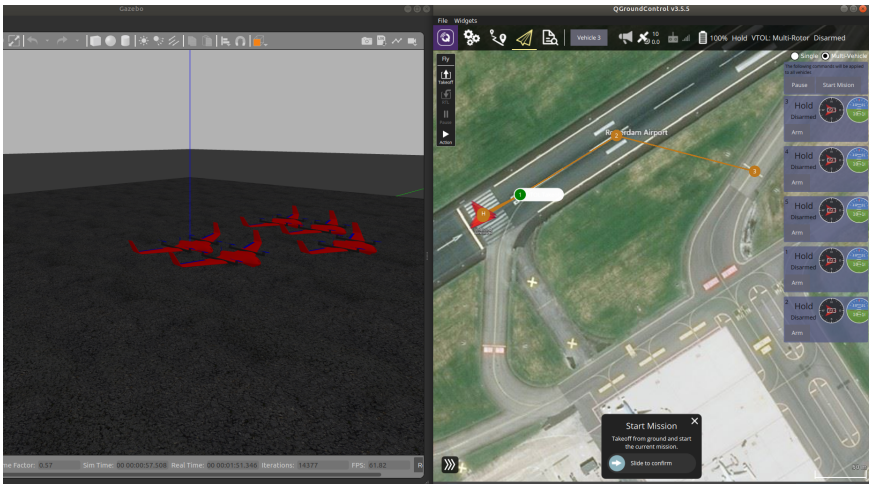


Figure 4.6: Launching 5 VTOL UAVs

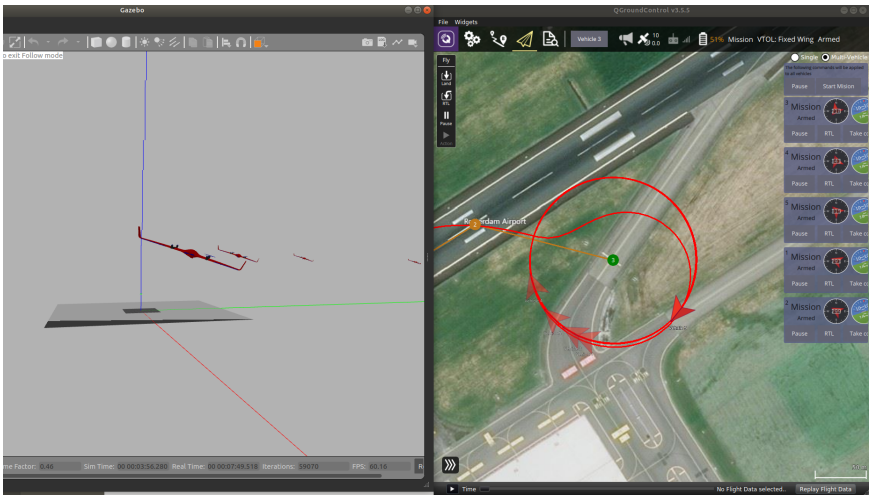


Figure 4.7: VTOL UAVs during flight and loitering

4.3.2 HITL environment for Multi-UAV

The simulation environment for multi-UAV is set by combining Sections: 4.2.2 and 4.3.1 i.e autopilot stack is build and uploaded onto the mutiple pixhawk boards which are then latched to Ubuntu machine having simulator through USB cables. Multiple ROS nodes are generated to publish and subscribe the data and command for each vehicle. This method was not successful as it caused a communication loss of more than 50% even for 2 vehicles.

4.4 HITL environment with Companion Computer

In this section, a brief procedure is discussed for simulation of UAVs having companion computer along with the PX4 hardware.
Companion Computer: It is an extra-brain to the onboard flight controller used to run computationally heavy algorithm or to perform additional tasks such as image recognition which simply cannot be handled by the controller board itself due to memory requirement. Companion Computers can be used to interface and communicate with Pixhawk flight controller using the mavlink protocol over a serial connection having 4 pin configuration (+5v, GND, Tx and Rx). By doing this the companion computer gets all the mavlink data produced by the autopilot (including GPS data) and can use it to make intelligent decisions during flight. Raspberry Pi 3 B+ model is used here as companion computer to achieve this configuration Figure 4.8 the complete architecture for this configuration is shown in Figure 4.9.

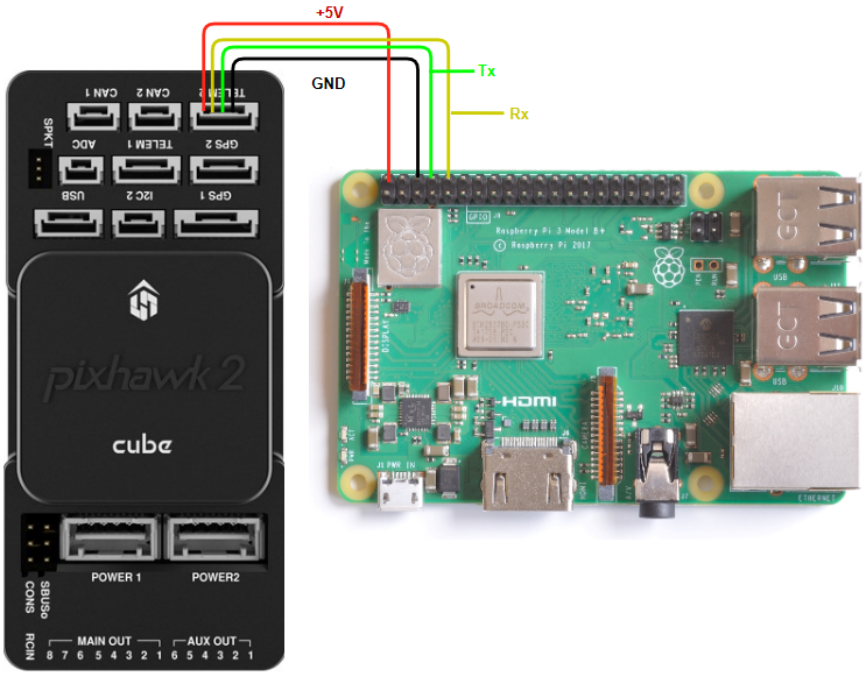


Figure 4.8: Companion Computer (RPI 3 b+) connection to Autopilot (Pixhawk2)

4.4.1 Serial Port Configuration in QGC

The serial (UART) ports on a Pixhawk board can be fully configured from QGC by accessing the parameters. The companion computer in discussion is configured on *TELEM 2* of Pixhawk board by selecting the parameters as given below:

- MAV_COMP_ID = 2: Mavlink component ID.

- MAV_2_CONFIG = TELEM 2: This parameter configures the port for serial communication for mavlink at instance 2; Here, it configures the serial port to run mavlink on port 102 (102 is is for TELEM 2).

- MAV_2_MODE = 2: Option 2 is used here for Onboard standard set of messages. This mavink mode defines the set of streamed messages and their sending rates for a companion computer,

- MAV_2_RATE = 921600 baud: This is a parameter for configuring the maximum sending rate for instance 2, the mavink streams in Bytes/sec. If this is set to 0, a value of /20 is used, which corresponds to half of the theoretical maximum bandwidth. The sending rate of the messages from each stream automatically lowers down, if the configured stream exceed the maximum defined rate.

- MAV_2_FORWARD = True: Setting this parameter to true enables mavlink message to forward for instance 2; i.eIf mode is enabled and either the point of reference is not the autopilot or it is broadcast, this configuration forwards the incoming mavlink messages to other mavlink ports. With help of this the QGC is able to talk to the companion computer that is connected to autopilot via mavlink protocol (on a different communication link than the ground control station).

It is necessary to set these parameters correctly and configure all the communication port from ground control station (QGC). A complete set of configuration parameters used in to configure ground control system and communication port are given in APPENDIX: B. QGroundControl also enables a user to automatically tune the controller (PID) gains through parameter access. A feature call autotune is also incorporated in QGC to automatically tune the gains.

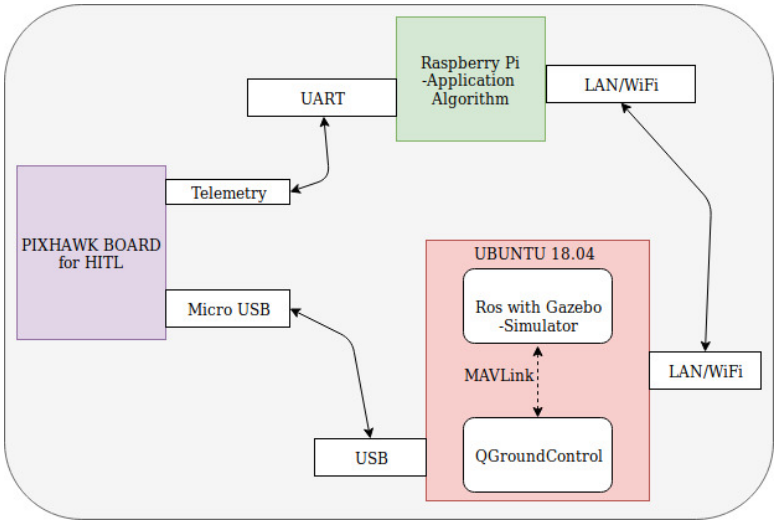


Figure 4.9: HITL Architecture with companion computer

The simulator is configured as explained and a HITL simulation is performed for the vehicle. For quad-rotor part of VTOL the simulation was successful as shown in Figure: 4.10 while it failed to run for fixed-wing part of the VTOL and can be explored in future work. The main reason for the failure in fixed wing mode is the interfacing of the companion computer with with autopilot and communication thereof.

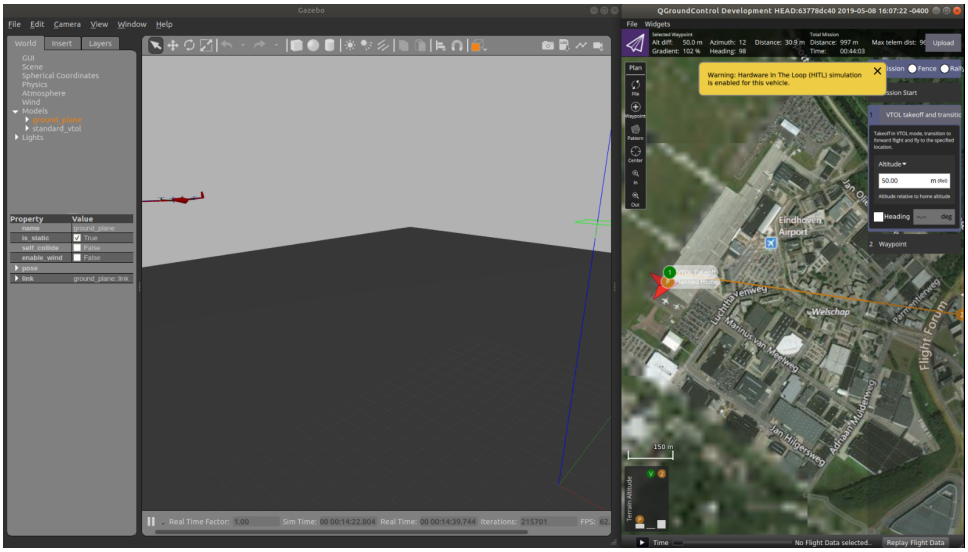


Figure 4.10: HITL simulation with companion computer

This concludes the simulator part discussion for this report. Till now, most of the aspects related to the thesis is covered in details. In next chapter, the various Matlab/Simulink simulations are discussed and the summary of result analysis is given.

Chapter 5

Simulations and Result Analysis

This chapter discusses the simulations performed for this work and the result analysis thereof. The chapter is organised as follows: First section of this chapter presents the simulation and describes the results of various formations. Next section discusses the simulations and results of formation without adaptation. In the third section, the simulation results of takeoff and loitering is shown. Finally, the simulations of formation switching by a flock of UAVs while being airborne is shown.

5.1 Simulation results of various formations

In line with most UAV path generation approaches, the path is composed of straight lines and orbits. Here, the simulations are performed for a flock of UAVs: To follow a straight line path and then the loiter around and orbit. One important point that needs mention is, that the low level controllers, implemented inside the autopilot stack (here Pixhawk/PX4) handles the kinematic constraints (roll/pitch/yaw and altitude); instead of the path following algorithm directly. This means that, if the radius of loitering (circular path) is chosen as 25 meters for all the UAVs; it should not violate the physical limits. Otherwise the UAVs will fail to follow the orbit due to maximum angle limit of the ailerons. It implies, the radius should not be too small to track the orbit by the autopilots. To be specific with respect to fixed-wing UAVs model, following constraints are prevalent in commercial fixed-wing UAVs: the elevators has an operating range of ± 15 degrees, the ailerons has an operating range ± 30 degrees and the rudder has an operating range ± 25 degrees. The UAVs starting position can be chosen arbitrarily which is implemented as initial conditions in the simulation. The initial attitude angles (roll/pitch/yaw) should be within the autopilot operating ranges, otherwise for aforementioned reasons, the autopilot will fail to stabilize the UAV models.

5.1.1 Path following in T formation

In Figure: 5.1 the simulations is shown for an inverted T-formation (T-formation) for the group of fixed-wing UAVs. For the aforementioned arrangement: the communication graph is shown in Figure: 2.7. For multi-UAV formation simulation, 4 different UAVs model and a Pinner UAV

model for (MRAC) is considered. The parameters and configuration of the UAVs is shown in Table: 2.3.

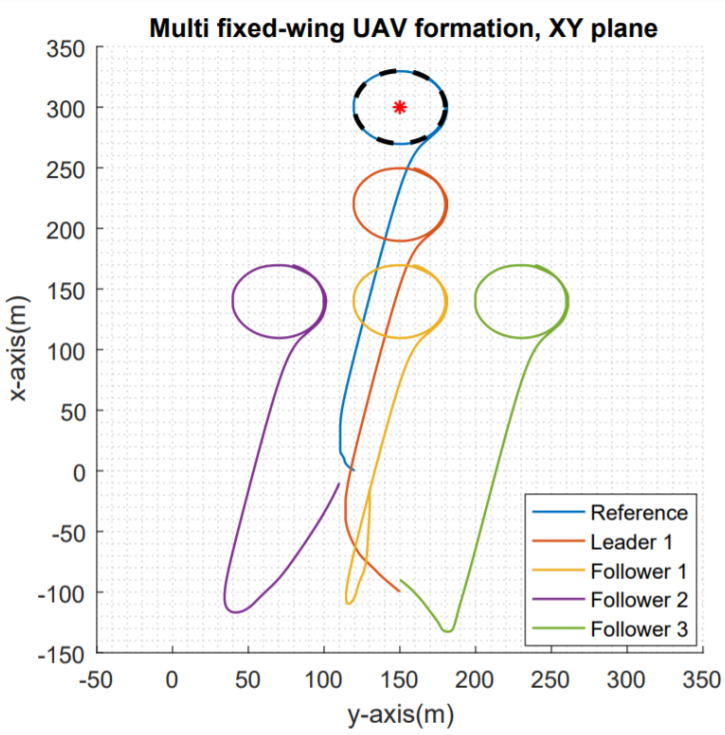


Figure 5.1: The flock of UAVs first follows a line and flies in T-formation and the loiters in an orbit maintaining the formation.

5.1.2 Path following in V formation

In Figure: 5.2 the simulations is shown for an inverted V-formation (V-formation) for the group of fixed-wing UAVs. For the aforementioned arrangement: the communication graph is shown in Figure: 2.8. For multi-UAV formation simulation, 4 different UAVs model and a Pinner UAV model for (MRAC) is considered. The parameters and configuration of the UAVs is shown in Table: 2.4.

5.1.3 Path following in Y formation

Finally, in Figure: 5.3 the simulations is shown for Y-formation for the group of fixed-wing UAVs. For the aforementioned arrangement: the communication graph is shown in Figure: 2.9. For multi-UAV formation simulation, 4 different UAVs model and a Pinner UAV model for (MRAC) is considered. The parameters and configuration of the UAVs is shown in Table: 2.5.

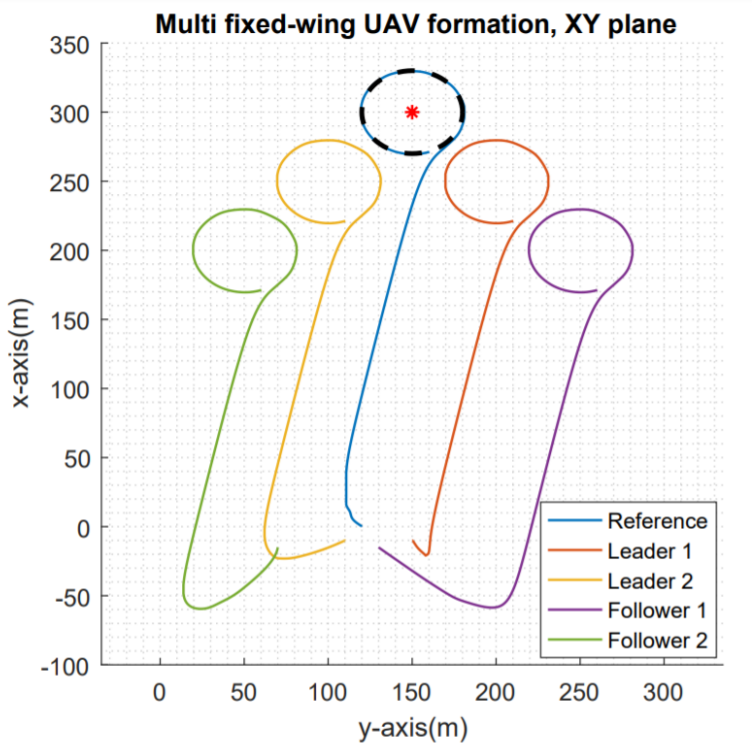


Figure 5.2: The flock of UAVs first follows a line and flies in V-formation and the loiters in an orbit maintaining the formation.

Result: 1

Here, the main noticeable point is all the UAVs in the formation has different masses and inertia: Despite the uncertainty the formation control for all varied arrangements (T-V-Y) is achieved. It clearly demonstrates the superiority and potency of the method proposed and applied in formation control.

Note: Advantage of the adaptive control law is that it allows the heterogeneous UAVs to homogenize and adopt to the same dynamics. It achieves the homogeneity by compensating for difference in inertia and mass through adaptive control correction. Throughout the formation control literature, it is found and well known fact that the homogeneous dynamics are a crucial feature for achieving proper coordinated motion [27], [41].

5.2 Path following without Adaptation

The simulation is set up in a similar way as explained for V formation or inverted V formation. It consists of two leader and two follower UAVs model, out of which one leader-follower pair ie. (Leader 1 and Follower 1) follows the adaptive control algorithm while the other leader-follower

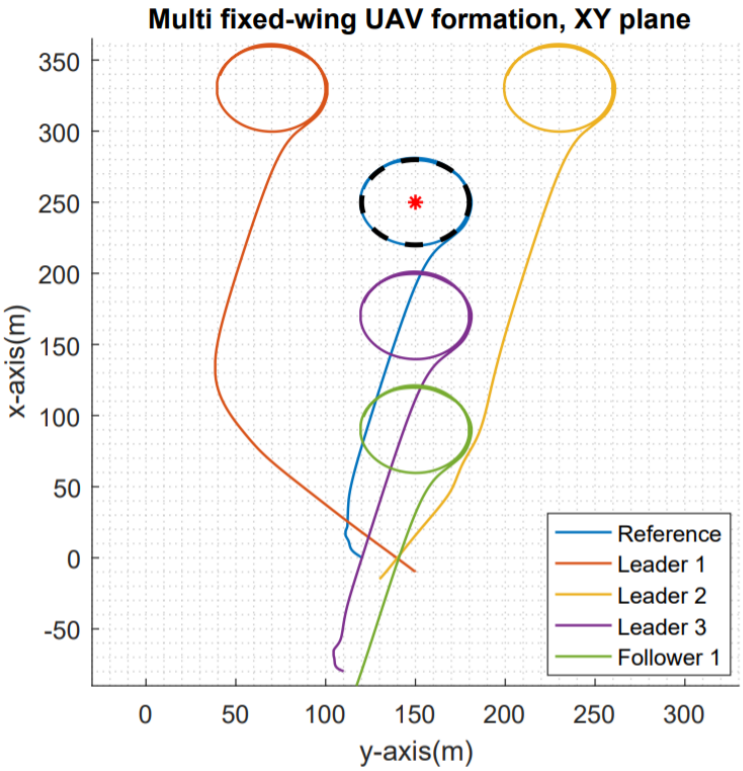


Figure 5.3: The flock of UAVs first follows a line and flies in Y-formation and the loiters in an orbit maintaining the formation.

(Leader 2 and follower 2) does not employ adaptive control. In other words, first pair consists of varying control gains while for the other pair gains are kept constant. This results into the failure for the second to adapt according to varying uncertain mass/inertia.

Result: 2

Here an important feature and relevance of adaptative control is highlighted: The simulation in Figure: 5.4 shows that in absence of adaption the two UAVs not employing adaptation cannot close the gap with respect to their predecessor and they eventually leave the formation. It can be noted from Table: 2.4 that the inertia and masses of the UAVs varies with a factor 10 and 5 respectively. Hence, this proves the uniqueness of and relevance of adaptive control algorithm by adapting to such an uncertain heterogeneous system. Evidently, this fact is proved here by utilizing software-in-the-loop simulation in case of the UAVs that the algorithm can handle uncertain inertia and mass and if such an adaptation is not present, it is quite difficult to work on formation control strategy.

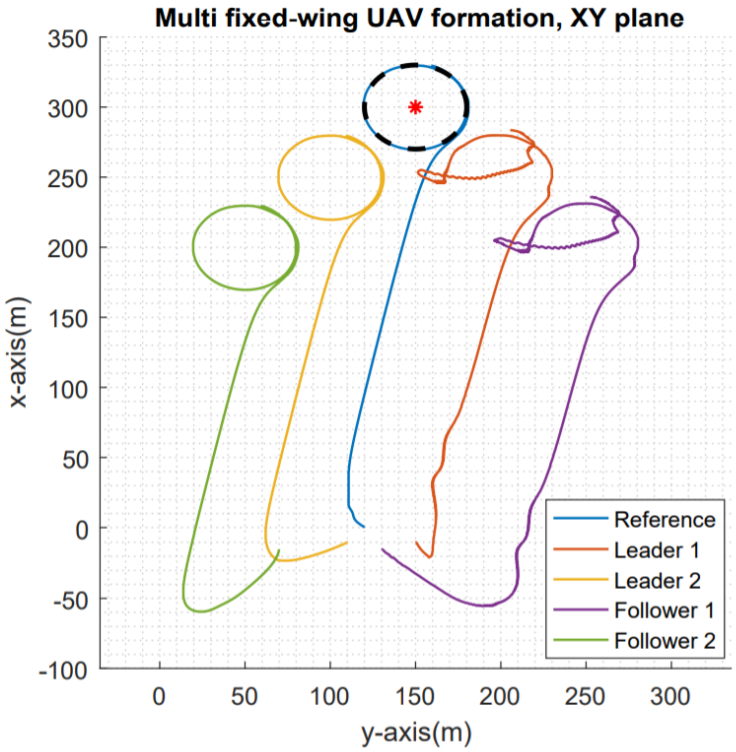


Figure 5.4: Simulation results highlighting a key feature of adaptive control algorithm: An Unsuccessful path following by Leader 1 and Follower 1 in absence of adaptive control laws. While Leader 2 and Follower 2 achieves the formation successfully by having an adaptive control algorithm.

5.3 Takeoff and Loitering

In this section, the Matlab simulation results for take-off and loitering of fixed-wing UAVs is shown utilizing the adaptive formation control law.

Simulation results in Figure: 5.5 shows the that five UAVs have took off from different positions on the ground and they are following a trajectory at different time instant. While the simulation results in Figure: 5.6 shows the UAVs have followed their trajectories and loitering around their respective loiter point without any formation. It is important to point out that, the formation control algorithm can also be instantiated now to fly in a formation.

5.4 Formation Switching

In this section, the simulation results are shown for a flock of UAVs: Took-off from different positions and starts to fly in T formation Figure: 5.7(a). Afterwards, it loiters around a given position maintaining the same formation Figure: 5.7(b) which then transitions from T to V-

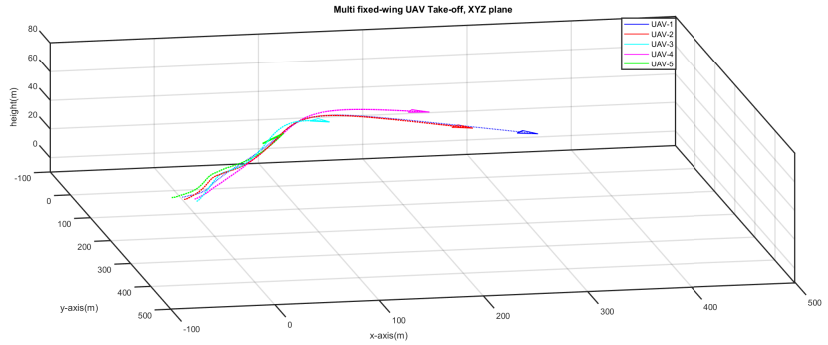


Figure 5.5: Simulation showing UAVS taking-off from different position on the ground at varied time instant

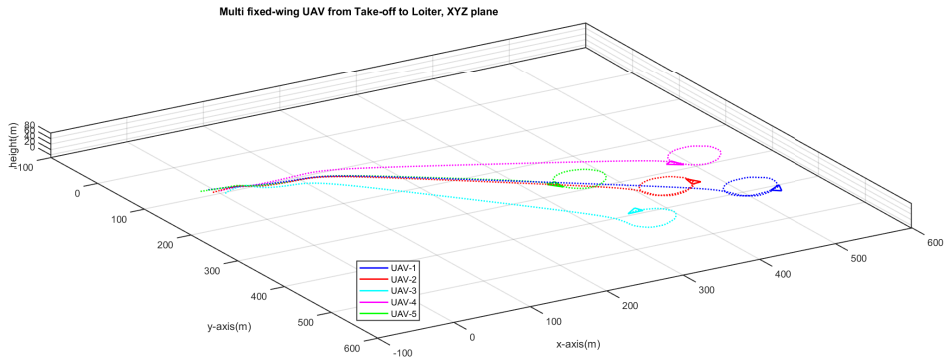
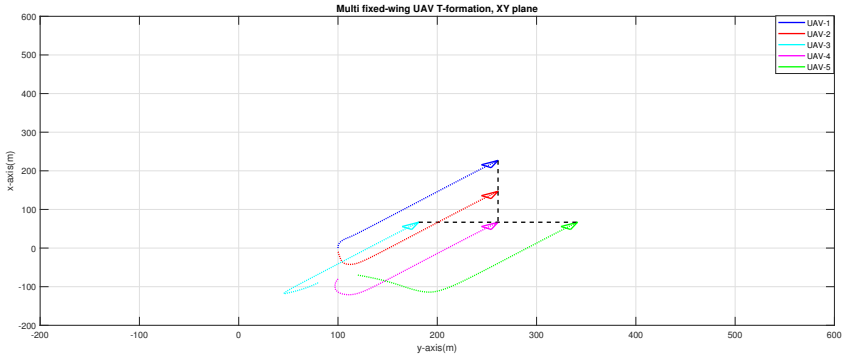


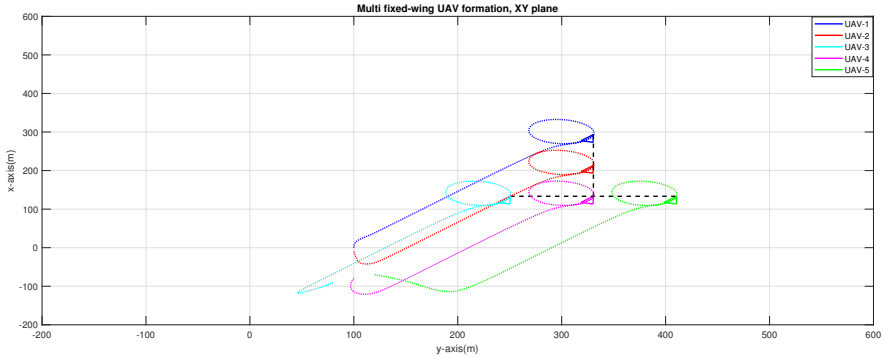
Figure 5.6: Five UAVs took-off from different positions on the ground at different time instant and reached their respective loiter point

formation as depicted in Figures: 5.7(c) and 5.7(d). As it can be seen trails of previous formation (T-formation) are removed to keep the picture neat. Figure: **??** show a completely transited formation and flock is flying towards the loitering point Figure: 5.8(b). In the next part; Figure: 5.8(c) shows a transition from formation V and Figure: 5.8(d) shows the transited Y-formation. Finally, the simulation ends with the Figure: 5.9 showing the flock reaching at the loitering point in Y-formation. As it can be seen trails of previous formation (T and V formations) are removed to keep the picture neat. At this point formation of all three formations while UAVs being airborne is shown.

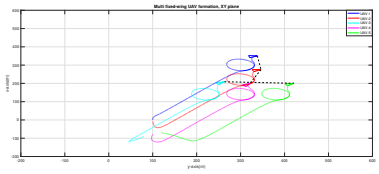
With this ends the simulations and results. In the next chapter the conclusions of this thesis work with some key points which can be addressed in future is discussed.



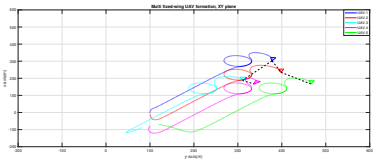
(a) UAVs taking-off from different positions and flying in T-formation



(b) Flock of UAVs loitering in T-formation

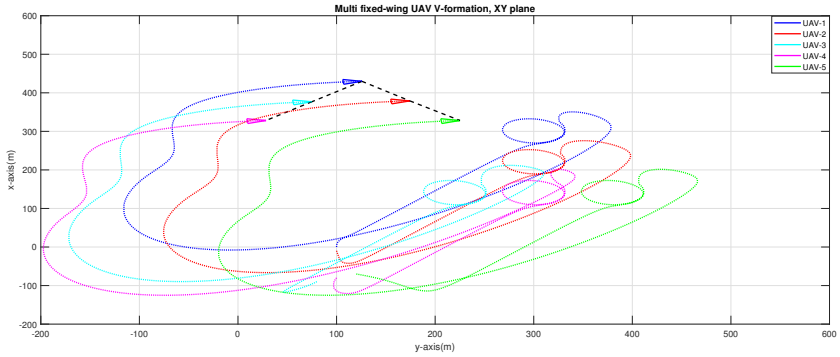


(c) Start of formation tansition from T to V-formation

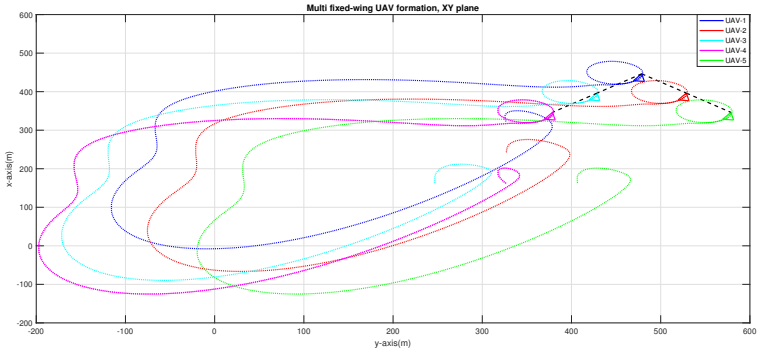


(d) Transited formation from T to V

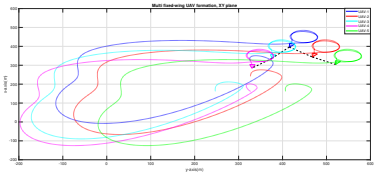
Figure 5.7: Figures showing the simulations of the UAVs at different stages: taking off from ground and reaching to the loiter point in T formation and started to transition and finally, transited to V formation



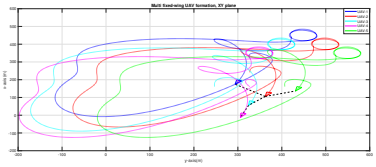
(a) UAVs completely transited and flying in V-formation



(b) Flock of UAVs loitering in V-formation



(c) Start of formation tansition from V to Y-formation



(d) Transited formation from V to Y

Figure 5.8: Figures showing the simulations of the UAVs at different stages: taking off from ground and reaching to the loiter point in V formation and started to transition and finally, transited to Y formation

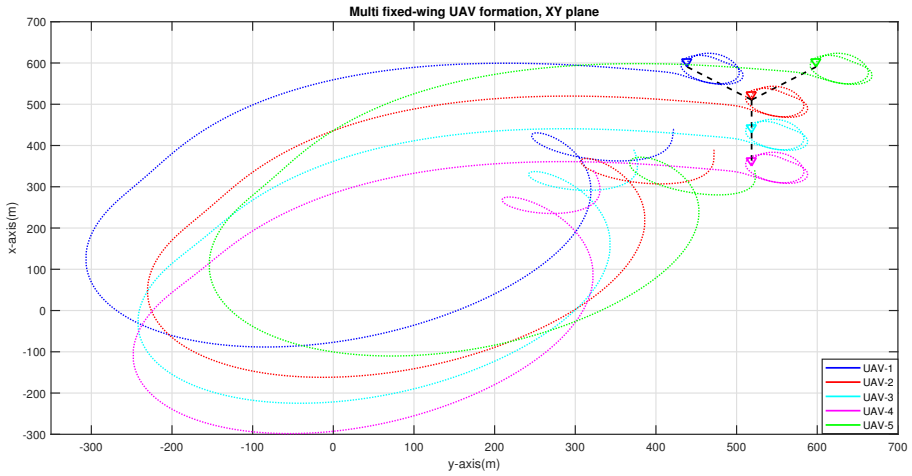


Figure 5.9: UAVs transited from V formation to Y-formation and loiters at the specified point.

Chapter 6

Conclusions and Future Work

In this chapter the conclusions are given based on the research work. The future possibility of the research that need to be carried out is given in the last and the final section of this chapter.

6.1 Conclusions

This work has discussed the adaptive formation laws with keeping focus on Fixed-wing unmanned aerial vehicles (UAVS) and its software-in-the-loop (SITL) implementation utilizing MATLAB/Simulink environment. As mentioned previously the two main focus areas this thesis posses; Various formation control algorithms with different gaps accompanying with the simulation of fixed-wing UAVs in Matlab/Simulink environment, in the presence of parametric uncertainties represented by uncertain mass and inertia. It also showed the failure of path following strategies in absence of adaptation. In the later half, Gazebo a 3D simulator for simulations in various environment software-in-the-loop (SITL), hardware-in-the-loop (HITL) and visualization of UAVs (VTOL) is developed with an aim that can be extended to real UAV scenarios.
This work also explains the software architectures and communication protocols that is utilized for configuring various hardwares needed to achieve (HITL) e.g HITL simulation configuration where PX4 is used as an autopilots to run low level controllers and Raspberry Pi as a companion computer to have an extra computation power that can be used while implementing formation algorithm.
The aspects that are tackled in relation to guidance and control for fixed-wing UAVs are: Matlab modelling of fixed-wing UAVs, vector field path following, adaptive formation control laws, software and hardware integration, PX4 autopilot low-level (pitch/roll/yaw) control, and last but not the least; software-in-the loop simulations showing various formation strategies. Finally, the simulator: Gazebo proved to be a powerful tool to depict real scenarios to perform simulations with the actual PX4 autopilot control structure of the Fixed-wing UAV (VTOL). For communication among the different component of vehicle and hardwares of real flights, MAVLink: the communication protocol was reverse engineered to and achieve the goal.

6.2 Future Work

This works opens up a completely new arena of research with humongous possibility. To start with, development and integration of the adaptive formation algorithm utilizing Gazebo as simulator. Secondly, to have a lower communication overhead a software-in-the-loop (SITL) simulation for the formation control laws utilizing PX4 low-level controllers. The task then can be taken and improvised for hardware-in-the-loop (HITL) simulation approaches, utilizing the simulator developed as part of this work. It is well known fact that, Ad-hoc networking allows to restructure a graph on the fly; hence utilizing technologies such as, Zigbee, Ad-hoc Wi-Fi networking or LoRaWAN etc. to setup communications in Switching Formation Topology and validate the effectiveness of various adaptive formations. Utilizing the work HITL with companion computer and PX4 hardwares can be extended for Fixed-wing part of UAV which then can be used to perform HITL showing various formations. Hardware in the loop Simulations performed as part of this thesis had communication overhead due to physics model running in simulator; the technique of Simulation-in-hardware (SIH) can be performed to achieve low communication overhead and increase the real time performance of the simulators. Finally, a simulation environment where companion computer is directly latched to the simulator again to reduce communication overhead and improve the performance of the simulator can be performed.

Bibliography

[1] PX4 Developers Community. [available : Online], https://dev.px4.io/v1.9.0/en/flight_stack/controller_diagrams.html.

[2] [available : Online], http://sdformat.org/spec?ver=1.6&elem=sdf. ©2019 Open Source Robotics Foundation.

[3] B. Min, J. Hong, and E. T. Matson. Adaptive robust control (arc) for an altitude control of a quadrotor type uav carrying an unknown payloads. In *2011 11th International Conference on Control, Automation and Systems*, pages 1147–1151, Oct 2011.

[4] Marconi Lorenzo Serrani Andrea Isidori, Alberto. *Robust Autonomous Guidance*. Springer-Verlag, 2003.

[5] E. Pastor, J. Lopez, and P. Royo. Uav payload and mission control hardware/software architecture. *IEEE Aerospace and Electronic Systems Magazine*, 22(6):3–8, June 2007.

[6] Patrick Doherty and Piotr Rudol. A uav search and rescue scenario with human body detection and geolocalization. In Mehmet A. Orgun and John Thornton, editors, *AI 2007: Advances in Artificial Intelligence*, pages 1–13, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[7] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka. Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics Automation Magazine*, 19(3):46–56, Sep. 2012.

[8] Muhammad Rosa, Simone Baldi, Ximan Wang, Maolong Lv, and Wenwu Yu. Adaptive hierarchical formation control for uncertain euler-lagrange systems using distributed inverse dynamics. *European Journal of Control*, 12 2018.

[9] Ilario Azzollini, Simone Baldi, and Elias Kosmatopoulos. Adaptive synchronization in networks with heterogeneous uncertain kuramoto-like units. pages 2417–2422, 06 2018.

[10] Wenwu Yu, Guanrong Chen, and Jinhu Lü. On pinning synchronization of complex dynamical networks. *Automatica*, 45(2):429 – 435, 2009.

[11] Wei Ren, Randal W. Beard, and Al W. Beard. Decentralized scheme for spacecraft formation flying via the virtual structure approach. *AIAA Journal of Guidance, Control, and Dynamics*, 27:73–82, 2003.

[12] Ruggero Carli, Fabio Fagnani, Paolo Frasca, and Sandro Zampieri. Gossip consensus algorithms via quantized communication. *Automatica*, 46(1), jan 2010v, issn =.

[13] L. Zhang, H. Gao, and O. Kaynak. Network-induced constraints in networked control systems—a survey. *IEEE Transactions on Industrial Informatics*, 9(1):403–416, Feb 2013.

[14] V. Borkar and P. Varaiya. Asymptotic agreement in distributed estimation. *IEEE Transactions on Automatic Control*, 27(3):650–655, June 1982.

[15] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics*, 9, 07 2012.

[16] Juan Carlos Rubio, Juris Vagners, and Rolf Rysdyk. *Adaptive Path Planning for Autonomous UAV Oceanic Search Missions*.

[17] P. B. Sujit, S. Saripalli, and J. B. Sousa. Unmanned aerial vehicle path following: A survey and analysis of algorithms for fixed-wing unmanned aerial vehicless. *IEEE Control Systems Magazine*, 34(1):42–59, Feb 2014.

[18] BingYu Zhou, Harish Satyavada, and Simone Baldi. Adaptive path following for unmanned aerial vehicles in time-varying unknown wind environments. *2017 American Control Conference (ACC)*, pages 1127–1132, 2017.

[19] Z.and Mettler B. Goerzen, C.and Kong. A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57(1):65, Nov 2009.

[20] 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(3):519–529, June 2007.

[21] A. Pedro Aguiar, Joao Hespanha, and Petar Kokotović. Performance limitations in reference tracking and path following for nonlinear systems. *Automatica*, 44:598–610, 03 2008.

[22] L. Furieri, T. Stastny, L. Marconi, R. Siegwart, and I. Gilitschenski. Gone with the wind: Nonlinear guidance for small fixed-wing aircraft in arbitrarily strong windfields. In *2017 American Control Conference (ACC)*, pages 4254–4261, May 2017.

[23] J. Yang, X. Wang, S. Baldi, S. Singh, and S. Fari. A software-in-the-loop implementation of adaptive formation control for fixed-wing uavs. *IEEE/CAA Journal of Automatica Sinica*, 6(5):1230–1239, Sep. 2019.

[24] Israel Lugo-C'ardenas, Gerardo Flores, and Rogelio Lozano. The mav3dsim: A simulation platform for research, education and validation of uav controllers. In *19th World Congress The International Federation of Automatic Control (IFAC 2014)*, pages 713–717, Cape Town, South Africa, Aug 2014.

[25] Gazebo. [available : Online], `http://gazebosim.org/projects`. ©2014 Open Source Robotics Foundation.

[26] S Fari. Guidance and control for a fixed-wing uav. MSc thesis, Politecnico di Milano, 2017.

[27] Simone Baldi and Paolo Frasca. Adaptive synchronization of unknown heterogeneous agents: an adaptive virtual model reference approach. *Journal of The Franklin Institute,Elsevier*, 356(2):pp.935–955, 2019. ff10.1016/j.jfranklin.2018.01.022.

[28] R. Olfati-Saber. *Nonlinear Control of Underactuated Mechanical Systems with Application to Robotics and Aerospace Vehicles*. PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2001.

[29] T. Kane and D. Levinson. *Dynamics, Theory and Applications*. McGraw-Hill, 1985.

[30] J. Huang W. Wang, C. Wen and Z. Li. Hierarchical decomposition based consensus tracking for uncertain interconnected systems via distributed adaptive output feedback control. *IEEE Transactions on Automatic Control*, 61(7):1938–1945, July 2016.

[31] Randal W. Beard and Timothy W. McLain. *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, Feb 2012.

[32] Hassan K. Khalil. *Nonlinear Systems*. Prentice-Hall, New Jersey, 2002.

[33] S. Fari, X. Wang, S. Roy, and S. Baldi. Addressing unmodelled path-following dynamics via adaptive vector field: a uav test case. *IEEE Transactions on Aerospace and Electronic Systems*, pages 1–1, 2019.

[34] Simone Baldi, Shuai Yuan, and Paolo Frasca. Output synchronization of unknown heterogeneous agents via distributed model reference adaptation. *IEEE Transactions on Control of Network Systems*, PP:1–1, 06 2018.

[35] L.F. Faleiro and A.A. Lambregts. Analysis and tuning of a 'total energy control system' control law using eigenstructure assignment. *Aerospace Science and Technology*, 3(3):127 – 140, 1999.

[36] Gazebo. [available : Online], `http://sdformat.org/`. ©2019 Open Source Robotics Foundation.

[37] Beat Küng Don Gagne Julian Oes Lorenz Meier, Hamish Willee. The standard communication protocol for drones; [available : Online], `https://mavlink.io/en/`.

[38] PX4 Developers Community. [available : Online], https://dev.px4.io/v1.9.0/en/simulation/hitl.html.

[39] Vladimir Ermakov vooon341 AT gmail DOT com. [available : Online], http://wiki.ros.org/mavros.

[40] PX4 Developers Community. [available : Online], https://dev.px4.io/v1.9.0/en/ros/mavros_installation.html.

[41] Youssef Abou Harfouch, Shuai Yuan, and Simone Baldi. An adaptive approach to cooperative longitudinal platooning of heterogeneous vehicles with communication losses. *IFAC-PapersOnLine*, 50(1):1352 – 1357, 2017. 20th IFAC World Congress.

Acronyms

AP: Access point

API: Application program interface

APF: Adaptive path following

EL: Euler Lagrange

GCS: Ground control system

GPS: Global positioning system

GUI: Graphical user interface

HITL/HIL: Hardware in the loop

LOS: Line of sight

MAV: Micro aerial vehicle

MRAC: Model reference adaptive control

NED: North east down

PC: Personal computer

PD: Proportional derivative

PI: Proportional and integral

QGC QGround control

ROS: Robot operating system

RPi: Raspberry Pi

SDF: Simulation description format

SITL: Software in the loop

SIH: Simulation in hardware

TCP: Transmission Control Protocol

TECS: Total energy control system

UART: Universal Asynchronous Receiver/Transmitter

UAV: Unmanned aerial vehicle

UDP: User Datagram Protocol

VF: Vector field

VTOL: Vertical take-off and landing

VTP: Virtual target point

WAP: Wireless access point

APPENDIX A

Building the simulator to launch VTOL model

- The first and foremost step is to have a linux machine and setup with all necessary software such as ROS melodic with Gazebo 9.0. `wiki.ros.org/melodic`, MAVlink and MAROS QGroundcontrol etc.
- Many issues may prompt depending on the configuration and version of OS and environment `https://dev.px4.io/master/en/simulation/` suggests how to acknowledge them.
- Clone from github `https://github.com/PX4/Firmware`; browse in folder Firmware to the launch file.
- Create a new launch file for single vehicle with vehicle type standard_vtol to generate simulation description format (sdf) model.
- In the same folder again create a new launch file for vehicle configuration and vehicle type standard_vtol for example; abcd_uav_mavros_model.launch.
- Build using the following commands:
- `DONTRUN=1 make px4_sitl gazebo_standard_vtol`
  It generates an sdf file which can be changed according to the need for example; the environment of simulation can be set such as SITL or HITL also the vehicle parameters itself can be configured.
- `source Tools/setup_gazebo.bash $(pwd) $(pwd)/build/px4_sitl_default.`
- `export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:$(pwd):$(pwd)/Tools/sitl_gazebo.`
  These two commands sets up the environment for simulation; Gazebo as simulator and ROS for communication nodes.
- `roslaunch px4 abcd_uav_mavros_model.launch`
  with this command pops up the Gazebo environment showing the vehicle model for simulation.

APPENDIX B

Building the simulator for VTOL model using companion computer

Hardware Connection

- Raspberry Pi containing the formation control algorithm is connected to the Pixhawk board using UART cable.
- The Pixhawk board is connected to Ubuntu machine through micro USB and USB respectively.

QGroundControl Parameterization.

In airframe first set the airfarme type `HIL Standard VTOL`. Then following parameters need to be configured.

- SYS_AUTOSTART = 1002.
- MAV_0_CONFIG = Disabled
- GPS_1_CONFIG = Disabled
- GPS_2_CONFIG = TELEM 2
  A reboot of the vehicle is needed here for further configuration of parameters.
- SER_TELE2_BAUD = 57600 8N1
- MAV_1_CONFIG = TELEM 2
- MAV_1_MODE = Onboard
- SER_TEL2_BAUD = 921600

After setting parameters, the procedure from APPENDIX: A shall be followed. Now, the commands to navigate or follow a path can be issued from Raspberry Pi using UART interface.