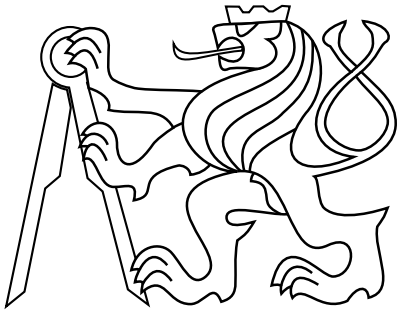
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Attitude-Aided Control of Leader-Follower Unmanned Aerial Vehicle Formation

Bachelor's Thesis

Josef Kahoun

Prague, April 2025

Study programme: Cybernetics and Robotics

Supervisor: Ing. Martin Jiroušek

Acknowledgments

First and foremost, I would like to thank my family and friends for their continuous support during my studies. I would also like to express my deepest gratitude to my supervisor, Ing. Marting Jiroušek, for his invaluable help, guidance, and willingness throughout the preparation of this thesis. Working under his supervision has greatly enhanced my skills and knowledge, for which I am truly grateful. Lastly, I would like to thank all the members of the Multi-robot Systems Group (MRS) for their assistance during the MRS camp, from which I gained many fond memories and valuable lessons in robotics.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kahoun Josef** Personal ID number: **516507**

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

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Attitude-Aided Control of Leader-Follower Unmanned Aerial Vehicle Formation

Bachelor’s thesis title in Czech:

Řízení formace bezpilotních letounů typu leader-follower s využitím údajů o náklonu

Guidelines:

The objective of this thesis is to design, implement and validate a control scheme for a leader-follower UAV formation that minimizes position tracking error by leveraging attitude measurements of the leader UAV. Traditional approaches based solely on position observations often suffer from delayed responses due to limited information about the leader’s dynamics. This thesis will address this limitation by developing an estimator-predictor that utilizes noisy pose measurements (position and orientation) to estimate the current state and predict the short-term trajectory of the leader UAV. The designed controller will use this prediction to improve the tracking performance of the follower UAV. The developed control scheme should be implemented in C/C++ within the ROS framework and tested using the MRS UAV system. The solution's effectiveness will be assessed in simulation and, if feasible, through real-world experiments.
Student tasks:
1) Study UAV control techniques, with a focus on the leader-follower problem. Familiarize yourself with Model Predictive Control (MPC) for UAVs. Review state estimation and prediction methods for dynamic systems.
2) Understand the ROS framework and the MRS UAV system.
3) Develop an estimator that fuses noisy pose measurements (position and orientation) to estimate the leader UAV’s current state. Extend the estimator or implement a separate module to predict the short-term future trajectory of the leader UAV.
4) Design and implement an MPC-based controller that utilizes the estimated/predicted state of the leader UAV. Choose between two control output options: (a) target collective thrust and attitude or (b) target collective thrust and attitude rate.
5) Conduct simulations to evaluate system performance. Compare tracking accuracy with and without leader attitude measurements. Analyze the robustness of the algorithm under imperfect data conditions.
6) If hardware is available, perform real-world flight tests to validate the algorithm.

Bibliography / sources:

1. J. Kim, S. A. Gadsden, and S. A. Wilkerson, "A comprehensive survey of control strategies for autonomous quadrotors," Canadian Journal of Electrical and Computer Engineering, vol. 43, no. 1, pp. 3–16, Winter 2020, doi: 10.1109/CJECE.2019.2920938.
2. H. Nguyen, M. Kamel, K. Alexis, and R. Siegwart, "Model predictive control for micro aerial vehicles: A survey," in Proc. 2021 European Control Conference (ECC), Delft, Netherlands, 2021, pp. 1556–1563, doi: 10.23919/ECC54610.2021.9654841.
3. M. Z. A. Rashid et al., "Comprehensive review on controller for leader-follower robotic system," Indian Journal of Geo Marine Sciences, vol. 48, no. 7, Jul. 2019. [Online]. Available: http://nopr.niscpr.res.in/handle/123456789/48866. Accessed: Jan. 30, 2025.
4. P. Shukla, S. Shukla, and A. K. Singh, "Trajectory-prediction techniques for unmanned aerial vehicles (UAVs): A comprehensive survey," IEEE Communications Surveys & Tutorials, 2024, doi: 10.1109/COMST.2024.3471671.
5. T. Baca, M. Petrlik, M. Vrba, et al., "The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles," Journal of Intelligent & Robotic Systems, vol. 102, no. 26, 2021, doi: 10.1007/s10846-021-01383-5.

Name and workplace of bachelor's thesis supervisor:

Ing. Martin Jiroušek Multi-robot Systems FEE

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

Date of bachelor's thesis assignment: **03.02.2025** Deadline for bachelor thesis submission: _____

Assignment valid until: **20.09.2026**

_____
prof. Dr. Ing. Jan Kybic
Head of department's signature

_____
prof. Mgr. Petr Páta, Ph.D.
Vice-dean´s signature on behalf of the Dean

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

_____
Date of assignment receipt

_____
Student's signature

FAKULTA ELEKTROTECHNICKÁ
FACULTY OF ELECTRICAL ENGINEERING
Technická 2
166 27 Praha 6

DECLARATION

I, the undersigned

Student's surname, given name(s): Kahoun Josef
Personal number: 516507
Programme name: Cybernetics and Robotics

declare that I have elaborated the bachelor's thesis entitled

Attitude-Aided Control of Leader-Follower Unmanned Aerial Vehicle Formation

independently, and have cited all information sources used in accordance with the Methodological Instruction on the Observance of Ethical Principles in the Preparation of University Theses and with the Framework Rules for the Use of Artificial Intelligence at CTU for Academic and Pedagogical Purposes in Bachelor's and Continuing Master's Programmes.

I declare that I used artificial intelligence tools during the preparation and writing of this thesis. I verified the generated content. I hereby confirm that I am aware of the fact that I am fully responsible for the contents of the thesis.

In Prague on 18.05.2025

Josef Kahoun
.....
student's signature

Abstract

This thesis presents the design, implementation, and evaluation of a control algorithm for a one-to-one leader-follower Unmanned Aerial Vehicle (UAV) formation, which exploits the attitude measurements of the leader UAV to minimize the tracking error. The UAV dynamics were studied, from which a linear mathematical model was derived. To estimate the states of the leader UAV, a Linear Kalman Filter (LKF) was designed, which utilized both the position and attitude measurements of the leader. Based on these estimates, a short-term trajectory predictor was built to forecast the leader's future states. The closed-loop control was achieved using a Quadratic Programming Model Predictive Control (QP-MPC), which utilized the predicted trajectory in order to improve the tracking performance. To assess the benefit of the attitude measurements, a baseline controller using only position measurements was also developed. The proposed system was rigorously tested both in simulation and real-world scenarios, demonstrating that incorporating attitude measurements significantly improves tracking performance. The algorithm's robustness to varying noise levels and update rates of the attitude measurements was also evaluated.

Keywords Unmanned Aerial Vehicle, Leader-Follower formation, Attitude measurements, Model Predictive Control, Kalman Filter, Trajectory prediction

Abstrakt

Tato práce se zabývá návrhem, implementací a vyhodnocením řídicího algoritmu pro formaci typu leader-follower (vůdce-následovník) s jedním bezpilotním letounem (UAV) v každé roli, který využívá měření náklonů vůdčího UAV ke snížení chyby sledování. Dynamika UAV byla studována a na jejím základě byl odvozen lineární matematický model. Pro odhad stavů vůdčího UAV byl navržen lineární Kalmanův filtr, který využíval jak měření polohy, tak měření náklonů vůdce. Na základě těchto odhadů byl vytvořen krátkodobý prediktor trajektorie, který předpovídal budoucí stavy vůdce. Zpětnovazební řízení bylo realizováno použitím prediktivního řízení založeného na kvadratickém programování (QP-MPC), které využívalo předpovězenou trajektorii za účelem vylepšení sledování. Pro posouzení přínosu měření náklonů byl vyvinut referenční regulátor, který pracoval pouze s měřeními polohy. Navržený systém byl důkladně otestován v simulaci i v reálném prostředí, přičemž bylo dokázáno, že využití měření náklonů výrazně zlepšuje přesnost sledování. Robustnost algoritmu vůči různým úrovním šumu a různým frekvencím měření byla rovněž vyhodnocena.

Klíčová slova Bezpilotní Prostředky, Formace typu vůdce-následovník, Měření náklonů, Prediktivní řízení na základě modelu, Kalmanův filtr, Predikce trajektorie

Abbreviations

GP Gaussian Process

HPIPM High-Precision Interior Point Method

LKF Linear Kalman Filter

LMPC Linear Model Predictive Control

LP Linear Programming

LQR Linear-Quadratic Regulator

LSTM Long Short-Term Memory

LTI Linear Time-Invariant

ML Machine Learning

MPC Model Predictive Control

MRS Multi-robot Systems Group

MSE Mean Squared Error

NMPC Nonlinear Model Predictive Control

QP Quadratic Programming

QP-MPC Quadratic Programming Model Predictive Control

RNN Recurrent Neural Network

ROS Robot Operating System

RTK-GPS Real-Time Kinematic Global Positioning System

SMC Sliding Mode Control

UAV Unmanned Aerial Vehicle

UKF Unscented Kalman Filter

Contents

1 Introduction

Multi-Agent Systems of Unmanned Aerial Vehicles (UAVs) have gained significant popularity in recent years due to their broad application in both civilian and military domains. In the military sector, UAVs are commonly employed for reconnaissance [1][2], surveillance [3], and search and rescue missions [4]. In the civilian field, drone swarms are largely being used in the entertainment industry [5], payload transportation [6], infrastructure mapping [7], etc.

However, the control of such systems poses additional challenges, particularly the risk of collisions between the UAVs. To mitigate this, various swarm control strategies have been developed. One of which is the widely adopted approach known as the Leader-Follower [8][9], where all UAVs (followers) track the trajectory of the designated leader UAV while consistently upholding a given relative position.

In order to further enhance the responsiveness and accuracy of the followers, researchers have been trying to predict the leader's future trajectory [10]. Provided with the information about the future development of the leader's trajectory, the follower UAV can implement preemptive actions, allowing for a much smoother and more accurate tracking. The task of trajectory prediction proves useful not only for control, but can also be useful for trajectory planning [11] and aerial defense [12][13] applications.

1.1 Motivation

For the follower UAV to track the leader UAV swiftly and accurately, predicting the leader's trajectory is highly advantageous. With access to the leader's future states, the follower can act preemptively, resulting in smoother and more responsive tracking.

In cases where the leader's trajectory is relatively simple—f.e. flying in a straight line at constant speed—knowledge of the UAV's current velocity is sufficient for an accurate prediction. However, during agile maneuvers, such as abrupt changes in flight direction, this information becomes inadequate. In such scenarios, both velocity and acceleration need to be known for a reliable prediction.

If only position is measured, acceleration becomes visible only after the motion has fully started, as it must be estimated through the second-order differentiation. On the other hand, if attitude data[1] is available, tilting behavior can be observed before a position change occurs. Since the attitude of the UAV is directly tied to its acceleration, this effectively enables us to detect a shift in the direction of movement much earlier.

If this presumption proves valid, incorporating attitude measurements into trajectory prediction could make control algorithms significantly more responsive to sudden changes in the leader's motion, and thus minimize the overall tracking error and delay.

1.2 Task Setup

The goal of this thesis is to design, implement, and evaluate the performance of a control system for a one-to-one Leader-Follower scheme, which implements noisy measurements of both the position and attitude of the leader UAV. The algorithm will control the follower's motion, while the leader will follow a predefined trajectory. The proposed control system will

[1] The knowledge about the UAV's rotation in the world.

be comprised of three main components: state estimation, short-term trajectory prediction, and the control algorithm. The whole pipeline is shown in Figure 1.1.

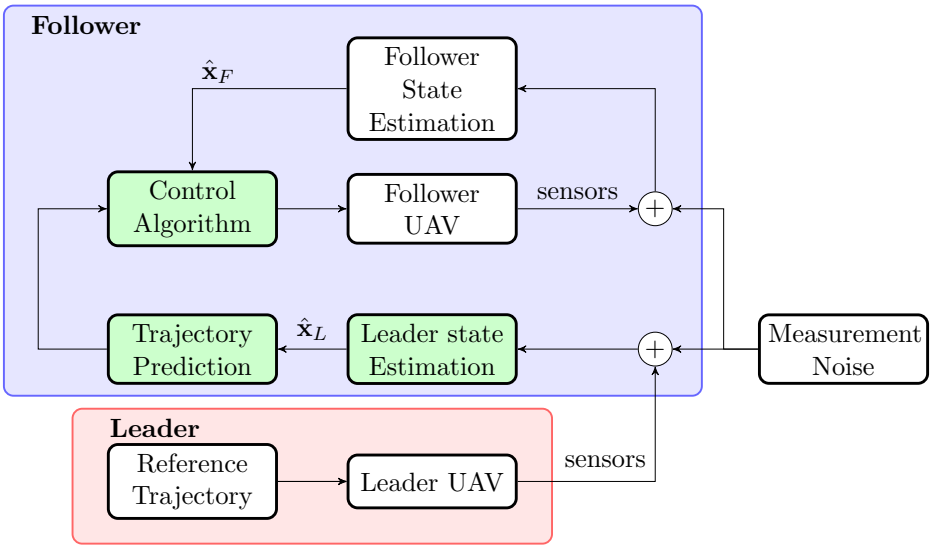


Figure 1.1: Outline of the system pipeline. The parts highlighted in green show the components to be implemented in this thesis. $\hat{\mathbf{x}}_L, \hat{\mathbf{x}}_P$ represent unspecified estimates of the leader/follower UAV states.

The attitude estimation of the leader is assumed to be provided externally, and therefore won't be implemented as a part of this thesis; however, the influence of the measurement noise and update rate of the readings will be thoroughly evaluated. The complete system is to be implemented in C++ and deployed on the UAV's onboard computer.

The structure of the thesis is as follows. In chapter 2, the linear model needed for control is derived together with the identification of unknown parameters. Chapters 3, 4, 5 cover the design of the leader state estimation, trajectory prediction, and the control algorithm, respectively. The implementation of the algorithms and tuning of the parameters are explained in chapter 6. Finally, the experiments evaluating the performance of the whole pipeline are presented in chapter 7.

1.3 Related Work

1.3.1 Control Algorithms

The leader-follower scheme is largely utilized in not just UAV swarms, but in almost every other multi-agent robotic system. As such, numerous control algorithms have been used and shown to be competent in the task of trajectory tracking in leader-follower formation [14].

Sliding Mode Control (SMC), a nonlinear control method which tries to drive the system state along a predefined 'sliding surface', has been used by researchers to control surface vessels [15] or UAVs [16].

From the realm of optimal control, approaches such as Linear-Quadratic Regulator (LQR), H_∞ [17] and Model Predictive Control (MPC) have been widely utilized in formation control. In [18], a two-layer distributed MPC controller has been designed by separating the UAV's dynamics into translation and rotation motions, thus reducing the computational load.

[19] introduced a combination of SMC and LQR controllers to successfully solve the multiple quadrotor formation problem.

Intelligent control schemes, such as neural networks [20] or reinforcement learning [21], have grown in popularity in recent years. Controllers based on fuzzy logic, a logic where truth values need not be binary, but may be any real number between 0 and 1, have also been utilized for both trajectory tracking [22] and formation control [23].

1.3.2 Trajectory Prediction

Basic methods for trajectory prediction include polynomial regression, which fits a curve to the UAV's past motion, using most commonly the Least-Squares Method, but this method is often prone to overfitting or underfitting. A simple alternative is to assume constant velocity or acceleration, but this approach is limited by the lack of information about the system dynamics.

More advanced techniques include the use of the Kalman Filters. In [24], the prediction step of the Linear Kalman Filter (LKF) is utilized to predict the future state of the UAV. A great advantage of this method is that it obtains covariance matrices as a side product of the prediction, providing probabilistic confidence in every predicted state.

A state-of-the-art approach involves Machine Learning (ML). The task of trajectory prediction is suitable for Recurrent Neural Networks (RNNs), which excel at processing sequential data, such as text, speech, or time series. In [25][26], a special type of RNN called Long Short-Term Memory (LSTM) is utilized to predict the future trajectory.

Another prominent ML approach is Gaussian Process (GP) Regression [27], which, unlike most ML models, returns a whole probabilistic distribution rather than the most probable value, making it much more suitable for capturing uncertainties.

Many other ML architectures have been explored. In [28], a Gated Cyclic Convolution Neural Network is utilized, a special type of RNN, much like LSTM. Originally developed for language processing, transformers [29] can also be adapted for trajectory prediction, and in [30], transformers were used to successfully predict trajectories of multi-agent UAV systems.

1.4 Contribution

This thesis focuses on improving the responsiveness of a follower UAV in a Leader-Follower configuration by incorporating attitude measurements of the leader UAV into trajectory prediction. The main contributions of this work are as follows:

(i) **Implementation of a control framework and a short-term trajectory prediction algorithm** that integrates both the position and attitude measurements of the leader UAV, as well as a reference framework which will not utilize the attitude measurements.
(ii) **Comparison of the performance of both controllers** in a simulated environment and in the real world, highlighting the impact of attitude information on the tracking accuracy.
(iii) **Evaluation of the effect of measurement noise and update rate of the attitude measurements** on the performance of the controller.

1.5 Mathematical Notation

In this thesis, the mathematical notation described in Table 1.1 is utilized.

x, k	scalar
$\mathbf{x}$, $\boldsymbol{\alpha}$	vector, pseudo-vector, or tuple
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{I}$	identity matrix
$\mathbf{x}^\top, \mathbf{X}^\top$	transposed vector/matrix
$\mathbf{x}_r$	$\mathbf{x}_r$ is *desired*, a reference
$x[n]/x_n$	x at the sample n
$\mathbf{x}_{k\|k-1}$	$\mathbf{x}$ at the sample k, conditioned by data obtained at the sample $k-1$
$\mathbf{R}_x(\alpha)$	rotation matrix coding rotation around x-axis by the angle α (likewise for y and z axes)
$\mathbf{R}_A^B$	rotation matrix from coordinate system A to coordinate system B
ϕ, θ, ψ	roll, pitch and yaw angles
$\mathcal{L}\{\cdot\}$	Laplace transform
$\dot{x}/\frac{\mathrm{d}x}{\mathrm{d}t}, \ddot{x}/\frac{\mathrm{d}^2x}{\mathrm{d}t^2}$	1^{st} and 2^{nd} time derivative of x
$\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}$	partial derivative of a function
$\Delta\mathbf{x}$	a change/difference/increment of $\mathbf{x}$
$a \wedge b$	a logical and b
$\mathbf{A}, \mathbf{B}, \mathbf{x}, \mathbf{u}$	LTI system matrix, input matrix, state vector and input vector
$\mathbf{c}^\top\mathbf{x}$	scalar product
$\nabla\mathbf{f}(\mathbf{x}) = \frac{\mathrm{d}\mathbf{f}(\mathbf{x})}{\mathrm{d}\mathbf{x}}^\top$	gradient of a function
$\nabla_\mathbf{x}\mathbf{f}(\mathbf{x},\mathbf{y}) = \frac{\partial \mathbf{f}(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}^\top$	partial gradient of a function

Table 1.1: Mathematical notation, nomenclature and notable symbols.

2 Model

2.1 Coordinate Systems

In order to unambiguously define the position and attitude of the UAV, we need to properly define the roll-pitch-yaw angles (denoted by ϕ, θ and ψ, respectively) and the related coordinate systems: the World frame (denoted by $\mathcal{W}$) and the Body frame (denoted by $\mathcal{B}$).

The origin of the World frame is fixed in space, with its x and y axes generating the horizontal plane, its z axis perpendicular to it, forming a right-handed coordinate system. The Body frame's origin is placed at the UAV's center of mass, its axes fixedly connected to the frame of the UAV.

To transform between these two systems, roll, pitch, and yaw—a special case of Tait-Bryan angles that follow the ZYX rotation order—are used. The Body frame is obtained by firstly rotating around the z-axis by an angle of ψ (yaw), then by rotating around the newly created y-axis by an angle of θ (pitch), and finally by rotating around the new x-axis by an angle of ϕ (roll). The coordinate frames are depicted in Figure 2.1. We can write this rotation utilizing the rotation matrices

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

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

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Following the given rotation order, the transformation from the Body frame to the World frame can be written as

$$\mathbf{R}_{\mathcal{B}}^{\mathcal{W}}(\phi, \theta, \psi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi). \tag{2.1}$$

2.2 Control Loop

Since controlling the individual rotors of a UAV is manually impossible for a normal human, most UAVs employ an inner control loop. This loop, implemented by the flight controller, takes as input the reference roll-pitch-yaw angles ϕ_r, θ_r, ψ_r[1] and the collective thrust T_r (the force along the z-axis of the Body frame), and adjusts the individual rotors accordingly to achieve the references.

[1] Another common implementation of the inner control loop takes as an input not the reference attitude, but rather reference attitude rates, but this approach will not be utilized in this thesis.

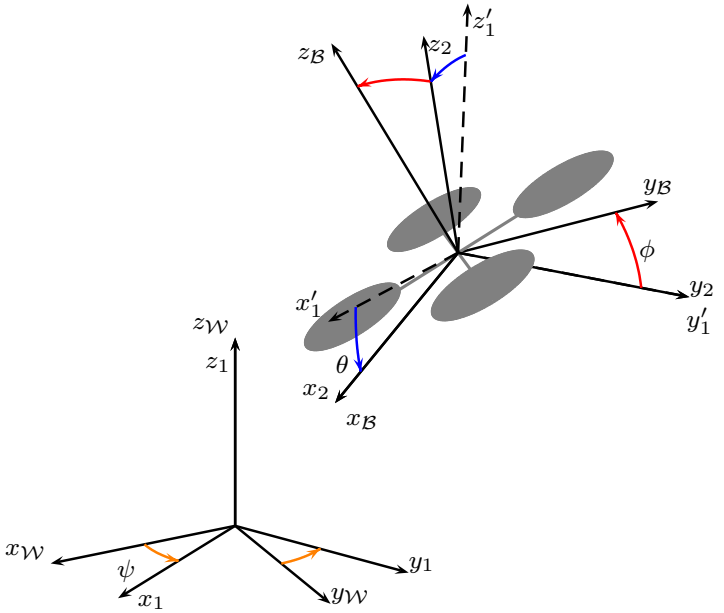


Figure 2.1: The UAV coordinate frames. The coordinate frame with the subscript $\mathcal{W}$ is the World frame, $\mathcal{B}$ is the body frame. Frames 1 and 2 are the coordinate frames obtained after applying the yaw and pitch rotations, respectively. The dashed frame (x_1', y_2', z_3') represents translated frame 1.

The outer control loop devised in the subsequent chapter controls the reference angles and thrust to achieve the given positional reference; therefore, the dynamics of the inner control loop must be taken into account when designing the outer control loop. The whole control scheme is shown in Figure 2.2

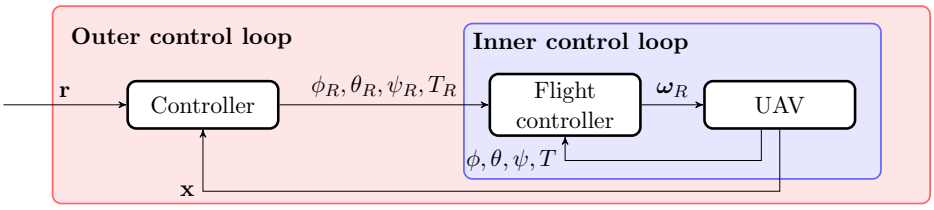


Figure 2.2: The whole control loop diagram. **r** represents a general reference, **x** are non-specific UAV states. $\boldsymbol{\omega}_R$ is the reference vector of angular velocities for each individual rotor.

2.3 UAV Dynamics

We derive the motion equations using Newton's second law of motion. The UAV generates thrust alongside the z-axis of its Body frame, while gravity acts on the UAV in the z-direction of the World frame. Other disturbances (e.g. drag) are neglected for the sake of simplicity.

Using the rotation matrix (2.1) to transform the thrust vector into the World frame, we

can write Newton's equations as

$$\begin{bmatrix} \ddot{x}^{\mathcal{W}} \\ \ddot{y}^{\mathcal{W}} \\ \ddot{z}^{\mathcal{W}} \end{bmatrix} = \frac{1}{m} \mathbf{R}_{\mathcal{B}}^{\mathcal{W}}(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}.$$

Upon substituting the rotation matrix, we can derive the following equations for the drone accelerations in the world frame

$$\begin{bmatrix} \ddot{x}^{\mathcal{W}} \\ \ddot{y}^{\mathcal{W}} \\ \ddot{z}^{\mathcal{W}} \end{bmatrix} = \frac{T}{m} \begin{bmatrix} \cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi \\ \sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi \\ \cos\theta \cos\phi \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix}. \tag{2.2}$$

2.4 Flight Controller Dynamics

As discussed, the inner control loop receives the reference attitude and thrust, and controls the individual rotors to achieve the given reference. Based on assumptions made in [31], we model the inner loop for each input as a first-order system:

$$\frac{\mathcal{L}\{\phi\}}{\mathcal{L}\{\phi_r\}} = \frac{K_1}{1+\tau_1 s},$$

$$\frac{\mathcal{L}\{\theta\}}{\mathcal{L}\{\theta_r\}} = \frac{K_2}{1+\tau_2 s},$$

$$\frac{\mathcal{L}\{\psi\}}{\mathcal{L}\{\psi_r\}} = \frac{K_3}{1+\tau_3 s},$$

$$\frac{\mathcal{L}\{T\}}{\mathcal{L}\{T_r\}} = \frac{K_4}{1+\tau_4 s},$$

where $\tau_{1,2,3,4}$ are time constants and $K_{1,2,3,4}$ are the gains.

For each first-order system (we will show the process for just the first one), we can easily find the differential equation in the time domain by multiplying both sides of the equation and using the inverse Laplace transform

$$\mathcal{L}\{\phi\}(1+\tau_1 s) = K_1 \mathcal{L}\{\phi_r\},$$

$$\phi(t) + \tau_1 \frac{\mathrm{d}\phi}{\mathrm{d}t} = K_1 \phi_r(t),$$

finally, by rearranging the terms and dividing the whole equation, the formula for the differential equation of the first-order system is obtained

$$\frac{\mathrm{d}\phi}{\mathrm{d}t} = \frac{K_1}{\tau_1}\phi_r(t) - \frac{1}{\tau_1}\phi(t). \tag{2.3}$$

2.5 The State Space Model

The state space model is an intuitive mathematical representation of the dynamics of the system, which can be generally written as

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.4}$$

where $\mathbf{x}$ is the state vector containing state variables, which provide information about the current properties of the system, and $\mathbf{u}$ is the input vector, containing current inputs for the system.

In the case of this thesis, we choose position, velocity, attitude, and thrust for the states[2]

$$\mathbf{x} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & \phi & \theta & \psi & T \end{bmatrix}^\top.$$

The control variables correspond to the inputs of the inner control loop, meaning the reference attitude and collective thrust

$$\mathbf{u} = \begin{bmatrix} \phi_r & \theta_r & \psi_r & T_r \end{bmatrix}^\top.$$

Combining the UAV dynamics (2.2), and the dynamics of the inner control loop (2.3), the overall nonlinear state space model for the UAV can be written as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{T} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \frac{T}{m}\left(\cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi\right) \\ \frac{T}{m}\left(\sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi\right) \\ \frac{T}{m}\left(\cos\theta \cos\phi\right) - g \\ \frac{K_1}{\tau_1}\phi_r - \frac{1}{\tau_1}\phi \\ \frac{K_1}{\tau_1}\theta - \frac{1}{\tau_1}\theta \\ \frac{K_1}{\tau_1}\psi_r - \frac{1}{\tau_1}\psi \\ \frac{K_1}{\tau_1}T_r - \frac{1}{\tau_1}T \end{bmatrix}. \tag{2.5}$$

2.6 Linearized Model

So far, only nonlinear equations have been derived. In order to utilize the linear controller used in this thesis, the nonlinear equations need to be linearized around an equilibrium point. Generally, the continuous Linear Time-Invariant (LTI) system can be written as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t), \\ \mathbf{y}(y) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t). \end{aligned} \tag{2.6}$$

In order to obtain the LTI model from our nonlinear equations, we first need to set an equilibrium point $(\mathbf{x}_0, \mathbf{u}_0)$, around which we will linearize the equations. We utilize the Taylor expansion of the general transfer function (2.4) around the equilibrium point

$$\dot{\mathbf{x}}(t) \approx \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) + \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{x}} \underbrace{(\mathbf{x}(t) - \mathbf{x}_0)}_{\Delta\mathbf{x}(t)} + \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{u}} \underbrace{(\mathbf{u}(t) - \mathbf{u}_0)}_{\Delta\mathbf{u}(t)}.$$

The equilibrium point, such as the name suggests, is chosen such that the system is in equilibrium, meaning that the states do not change. Therefore, the time derivatives of the states are zero, i.e. that $\mathbf{f}(\mathbf{x}_0, \mathbf{u}_0) = 0$. Upon substituting $\mathbf{A} = \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{x}}$ and $\mathbf{B} = \frac{\partial \mathbf{f}(\mathbf{x}_0, \mathbf{u}_0)}{\partial \mathbf{u}}$, we can write the LTI system as

$$\dot{\mathbf{x}} = \mathbf{A}\Delta\mathbf{x}(t) + \mathbf{B}\Delta\mathbf{u}(t).$$

The absolute value of the state vector at any given time can be obtained as $\mathbf{x}(t) = \mathbf{x}_0 + \Delta\mathbf{x}(t)$.

[2]Here on out, the position and velocity are in the World frame coordinates, therefore, no superscripts are used.

2.6.1 Linearization

The linearized state space matrices $\mathbf{A}, \mathbf{B}$ are computed by using the standard linearization method:

$$\mathbf{A} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x}_0,\mathbf{u}_0)}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x}_0,\mathbf{u}_0)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x}_0,\mathbf{u}_0)}{\partial x_1} & \cdots & \frac{\partial f_n(\mathbf{x}_0,\mathbf{u}_0)}{\partial x_n} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} \frac{\partial f_1(\mathbf{x}_0,\mathbf{u}_0)}{\partial u_1} & \cdots & \frac{\partial f_1(\mathbf{x}_0,\mathbf{u}_0)}{\partial u_m} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x}_0,\mathbf{u}_0)}{\partial u_1} & \cdots & \frac{\partial f_n(\mathbf{x}_0,\mathbf{u}_0)}{\partial u_m} \end{bmatrix},$$

where n is the number of states, m is the number of inputs, $(\mathbf{x}_0, \mathbf{u}_0)$ is the equilibrium point, and $f_{1,\dots,n}$ are the individual components of the nonlinear transition model.

To find the equilibrium point, we set the state equations (2.5) equal to zero, resulting in multiple periodical possible solutions. Since we, however, require only one model, we select the following equilibrium conditions:

$$T = mg \quad \wedge \quad \phi = 0 \quad \wedge \quad \theta = 0 \quad \wedge \quad \psi = 0. \tag{2.7}$$

These conditions were chosen based on symmetry around the Tait-Bryan angles, which makes this point the most suitable for our application.

Upon calculating the partial derivatives and substituting the chosen equilibrium point, we obtain the following sparse matrices

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{m} \\ 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\tau_4} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{K_1}{\tau_1} & 0 & 0 & 0 \\ 0 & \frac{K_2}{\tau_2} & 0 & 0 \\ 0 & 0 & \frac{K_3}{\tau_3} & 0 \\ 0 & 0 & 0 & \frac{K_4}{\tau_4} \end{bmatrix}.$$

2.7 Yaw Constraint

If the yaw were to change significantly, the linearized equations would become inaccurate. To prevent this, we introduce an additional constraint on our system, and that is that the yaw angle is always equal to zero. This means that the controller outputs 0 for the reference yaw angle at all times.

While the yaw constraint may seem limiting, in our case, the follower relies on measuring the relative position of the leader. Therefore, fixing the yaw to be constant is an adequate choice, as it simplifies the measurement process. The yaw could be set to an arbitrary value, but for the sake of simplicity, we chose zero.

2.8 Discretization

Since most of the controllers operate in discrete time, we need a way to transform the previously derived continuous-time model to discrete time. The simplest way to do so is to utilize the Forward (Explicit) Euler method. Generally, this method can be written as

$$\mathbf{x}[k+1] = \mathbf{x}[k] + T_s \mathbf{f}(\mathbf{x}[k], \mathbf{u}[k]),$$

where $\mathbf{f}$ is the transition function, and T_s is the sampling time.

Applying this to the case of a linear state model, we can substitute for $\mathbf{f}$

$$\begin{aligned}\mathbf{x}[k+1] &= \mathbf{x}[k] + T_s(\mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k]) = \underbrace{(\mathbf{I} + T_s\mathbf{A})}_{\mathbf{A}_d}\mathbf{x}[k] + \underbrace{T_s\mathbf{B}}_{\mathbf{B}_d}\mathbf{u}[k] \\ &= \mathbf{A}_d\mathbf{x}[k] + \mathbf{B}_d\mathbf{u}[k],\end{aligned}$$

receiving the system matrices in the discrete time as $\mathbf{A}_d = \mathbf{I} + T_s\mathbf{A}$ and $\mathbf{B}_d = T_s\mathbf{B}$.

2.9 Identification

Identification of the parameters was done by sending a step reference to the inputs of the inner controller and collecting the data. Subsequently, a first-order step response was iteratively fitted by hand by changing the gain and time constant parameters.

For the angle identification, a step reference of height 0.5 (to not recede far away from the equilibrium point) was used, and for the thrust, a value of 1 was sent. For the attitude identification, we read directly the ground truth measured angles; for the thrust, we read the acceleration in the z-axis a_z. Since all Tait-Bryan angles were set to zero, the current thrust could be obtained simply by $T(t) = ma_z(t)$, where m is the mass of the UAV.

The behavior of the roll and pitch subsystems was presumed to be equivalent, which proved true in the simulations. We settled on the following parameters: $K_1 = K_2 = 1$, $\tau_1, \tau_2 = 0.15$ for the roll and pitch subsystems[3], and $K_4 = 1$, $\tau_4 = 0.1$ for the thrust subsystem. The comparison of the step responses of the real and fitted subsystems is shown in Figure 2.3.

[3]The identification of the yaw subsystem was not conducted, as it is not needed due to our introduced constraint.

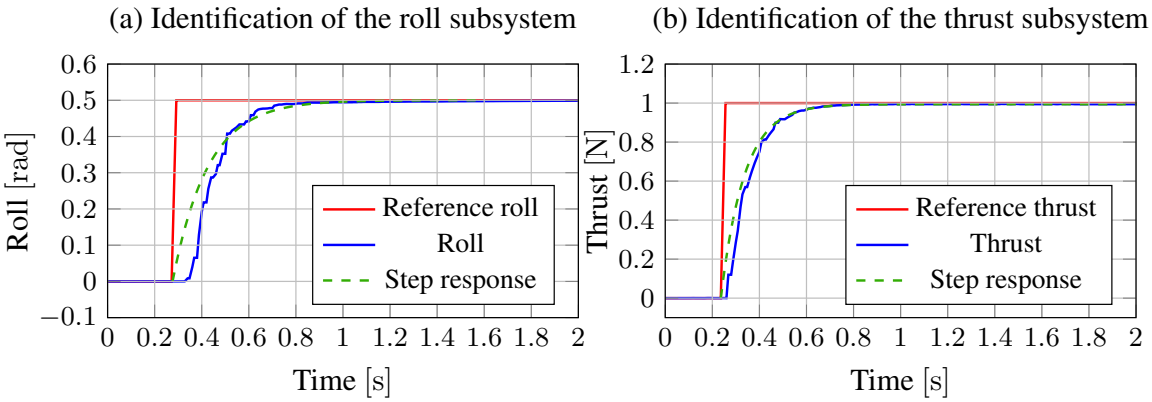


Figure 2.3: Identification of the inner control loop parameters for the roll (a) and thrust (b).

3 Leader State Estimation

To follow the leader's trajectory, the follower UAV requires information about the leader's state to utilize in the feedback control loop. This data is typically obtained through the use of different sensors depending on the application. However, direct sensor data is inherently noisy and differs from the ground truth—the real values of the physical quantities being measured. Therefore, a method for filtering out the noise from the acquired data is necessary.

3.1 Linear Kalman Filter

The LKF is a specialized type of estimator used in the control loop. It is a recursive algorithm that estimates the hidden states of the system by minimizing the Mean Squared Error (MSE) of the estimates.

3.1.1 Stochastic Model

The LKF assumes a discrete-time linear model defined by 2 equations: the state transition and the observation.

(i) State Transition (Process Model)
The state transition equation can be written as

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u}_k + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k),$$

where $\mathbf{x}_k, \mathbf{u}_k$ are the state and input vectors at time k, respectively, $\mathbf{A}, \mathbf{B}$ are the state and input matrices, and $\mathbf{w}_k$ is the process noise.

(ii) Observation (Measurement Model)
The measurement model equation is described by

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \quad \mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k),$$

where $\mathbf{z}_k$ is the measured system output at time k, $\mathbf{H}$ is the system output matrix, which maps the states to the outputs/measurements, and vector $\mathbf{v}_k$ is the measurement noise.

Noticeably, the LKF model is almost identical to the LTI model defined in (2.6)[1], with the addition of the modeled noises. Both noises are assumed to be white, i.e. all frequencies have the same intensity, and to be sampled from the Gaussian distribution. Specifically, distributions for the process and measurement noise are zero-mean, with the covariance given by covariance matrices $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively. In real-world applications, the noises rarely adhere to the white Gaussian specification. However, approximating them as such proves to be sufficient in most practical cases.

3.1.2 The Algorithm

The LKF computes the estimated state vector $\hat{\mathbf{x}}_k$ and the associated estimated covariance $\mathbf{P}_k$, which models the uncertainty of each state estimate. Each iteration consists of two steps:

[1] The feedthrough matrix $\mathbf{D}$ is considered to be zero, as is most often the case.

(i) Prediction (Time Update)
The prediction step gives apriori estimation of the state and covariance in the next timestep, and is given by two equations,

$$\begin{aligned} \hat{\mathbf{x}}_{k+1|k} &= \mathbf{A}\hat{\mathbf{x}}_{k|k} + \mathbf{B}u_k, \\ \mathbf{P}_{k+1|k} &= \mathbf{A}\mathbf{P}_{k|k}\mathbf{A}^\top + \mathbf{Q}_k. \end{aligned} \tag{3.1}$$

$\hat{\mathbf{x}}_{k+1|k}$ is the state vector estimation for time $k+1$ conditioned by the information obtained from measurements at step k, likewise, the $\hat{\mathbf{x}}_{k|k}$ is the state vector estimation for time k using the data available at time k, $\mathbf{u}_k$ is the system control input at current step k, and $\mathbf{Q}_k$ is the process noise covariance matrix at time k. The same math notation used for the state estimates concerns the covariance estimates $\mathbf{P}$.

(ii) Correction (Measurement Update)
The correction step uses the current acquired noisy measurements to correct the apriori estimation of both the state vector and covariance. Firstly, the Kalman Gain $\mathbf{K}_{k+1}$ is calculated from the the current covariance estimation. It tells the filter how much to trust the new measured data, and how much of it to use to correct the apriori state and covariance estimates. Then, the state estimation for the next timestep $\hat{\mathbf{x}}_{k+1}$ is obtained with the measured data $\mathbf{z}_{k+1}$, and finally the covariance $\mathbf{P}$ is updated.

$$\begin{aligned} \mathbf{K}_{k+1} &= \mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^\top \left(\mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}_{k+1}^\top + \mathbf{R}_{k+1}\right)^{-1}, \\ \hat{\mathbf{x}}_{k+1|k+1} &= \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\left(\mathbf{z}_{k+1} - \mathbf{H}_{k+1}\hat{\mathbf{x}}_{k+1|k}\right), \\ \mathbf{P}_{k+1|k+1} &= (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\mathbf{P}_{k+1|k}. \end{aligned} \tag{3.2}$$

3.2 LKF Model

The need for a LTI model is apparent from the equations (3.1) and (3.2), however, the model derived in chapter 2 cannot be used directly, as it requires the leader's control inputs, which are unavailable to the follower UAV. A new model must therefore be constructed.

3.2.1 State Definition

In the model from chapter 2, the control inputs were the reference roll, pitch, and yaw angles, and the reference thrust. One approach would be to take out the inner control loop and use just the current measured angles as an input. However, the mathematical model of the LKF cannot model input noise, which is expected to be non-negligible. We therefore treat the angles and thrust as states and refine them only in the update step as a part of our measurement. We also add the angular velocities to the state variables, yielding the resulting state vector $\mathbf{x} = [x, y, z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_\phi, \omega_\theta, \omega_\psi, T]^\top$ with a total of 13 state variables.

3.2.2 Transition Model

Now, the transition model can be written, mostly similar to the one from equation (2.5), with the added influence of angular velocities. We cannot obtain the differential equations for the angular velocities and the thrust, as we do not have a model for them. We therefore make the assumption that these states stay constant (the differential equations are zero), with the associated modeling error captured in the process noise. This results in the following nonlinear

differential equations

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\omega}_\phi \\ \dot{\omega}_\theta \\ \dot{\omega}_\psi \\ \dot{T} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \frac{T}{m}\left(\cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi\right) \\ \frac{T}{m}\left(\sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi\right) \\ \frac{T}{m}\left(\cos\theta\cos\phi\right) - g \\ \omega_\phi \\ \omega_\theta \\ \omega_\psi \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.3}$$

3.2.3 Linearized System Matrices

By linearizing around the equilibrium point from (2.7), we obtain the following linear system matrix

$$\mathbf{A}_{LKF} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{m} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.4}$$

The input matrix $\mathbf{B}_{LKF}$ is zero since there are no control inputs. The output matrix $\mathbf{H}$ is constructed based on the available measurements, which consist of the position and orientation of the UAV, creating

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.5}$$

Since we linearized around the equilibrium point (2.7), the differential equations for the horizontal velocities assume a constant thrust $T = mg$, and therefore a constant height. As a result, during tuning and experiments, we only use trajectories flown at a constant height.

4 Trajectory Predictor

When dealing with agile movement, setting the follower only the current reference position may be shortsighted, as by the time it reaches the given position, the leader's reference may have already changed. Obtaining information about the future development of the leader's trajectory can improve the performance of the tracking algorithm and reduce both the tracking error and delay in the follower's motion.

Furthermore, this approach allows to fully utilize the attitude measurements, as their main benefit lies in providing better estimates of the leader's acceleration. As a result, predicting the trajectory during sudden changes in the direction of movement becomes more reliable.

4.1 LKF Prediction

As discussed earlier, the LKF provides an estimate of the current states of the leader UAV. We can extend this by using the prediction step of the LKF (3.1) to iteratively predict the UAV's future states. Starting from the current state estimate $\hat{\mathbf{x}}[t]$ and corresponding covariance matrix $\mathbf{P}[t]$, we apply the prediction step and obtain the resulting state prediction $\hat{\mathbf{x}}[t+1]$ and covariance matrix $\mathbf{P}[t+1]$. The prediction step is ultimately repeated N times, where N is the desired length of the prediction. This process yields an array of state estimates and associated covariance matrices, each one specified at time kT_s, where k is the index of the prediction, and T_s is the sampling period of the LKF model. It is also apparent that the total time span of the prediction can be computed as NT_S. The covariance matrices $\mathbf{P}[t], \ldots, \mathbf{P}[t+N]$ may seem just as a side product. However, each matrix can be used to assess the confidence in the corresponding reference position—that is, the probability that the predicted position will actually be reached by the leader. The whole pseudo-algorithm for the trajectory prediction is shown in Algorithm 1.

Algorithm 1 Trajectory prediction

$T \leftarrow$ new array of size $N+1$ ▷ N is the number of prediction steps
$\mathbf{x}_{\text{tmp}} \leftarrow \hat{\mathbf{x}}$ ▷ Initialize with the current LKF estimation
$\mathbf{P}_{\text{tmp}} \leftarrow \mathbf{P}$
$T[0] \leftarrow (\mathbf{x}_{\text{tmp}}, \mathbf{P}_{\text{tmp}})$ ▷ We want to save the current estimation
for $i = 1$ **to** N **do**
    $\mathbf{x}_{\text{tmp}}, \mathbf{P}_{\text{tmp}} \leftarrow$ PREDICTION_STEP$(\mathbf{x}_{\text{tmp}}, \mathbf{P}_{\text{tmp}})$
    $T[i] \leftarrow (\mathbf{x}_{\text{tmp}}, \mathbf{P}_{\text{tmp}})$
end for
return T

4.2 Nonlinear Prediction

We can further improve prediction accuracy by using the nonlinear equations (3.3) directly instead of relying on the linear approximation. However, by predicting using just the nonlinear equations, we lose the covariance matrices, thereby losing the valuable information about the uncertainty of the predictions. We therefore utilize the LKF-based prediction to

generate covariance matrices, while replacing the predicted state means with those obtained from the nonlinear equations.

It's worth noting that a nonlinear variant of the Kalman Filter, namely the Unscented Kalman Filter (UKF) [32] has been implemented and tested for both estimation and trajectory prediction. While the LKF and the UKF proved almost equal in the estimation domain, the prediction of the trajectory utilizing the UKF was more accurate. However, it was also more computationally demanding—exceeding even the MPC computation time—which ultimately led to instability within the controller. We therefore adopted a hybrid approach: generating the covariance matrices through the LKF, while the state predictions are computed using the nonlinear equations. As a result, the predicted covariance matrices are only approximate, but they are sufficiently accurate for our purposes.

5 Controller

Having the predicted trajectory, we require a controller that can fully exploit this information. Ultimately, MPC is an ideal choice, as its ability to plan and adjust control inputs based on future references sets it apart from other control algorithms.

MPC belongs to the field of optimal control. In optimal control, there are two primary approaches to the problem, the *indirect approach* and the *direct approach*. The *indirect approach* seeks to derive a fixed, analytical controller. In contrast, the *direct approach* optimizes the control actions directly, which requires an online optimization, one that is far more computationally expensive but provides more versatility than the *indirect approach*. MPC stands out as one of the most prominent examples of the *direct* approach.

5.1 Discrete-Time Optimal Control Problem on Finite Horizon

The basis of MPC lies in solving the *discrete-time optimal control problem on finite horizon*. In this problem, a discrete-time system is controlled by a feedforward controller on a finite horizon N. The goal of the controller is to obtain a sequence of control actions $\mathbf{u}[0], \mathbf{u}[1], \ldots, \mathbf{u}[N]$ that minimizes a given function $J(\cdot)$, often called the *cost function*. The cost function serves as a way to qualify the performance of the controller based on the sequence of the control actions and the states of the system, i.e.

$$J\left(\mathbf{u}[t], \ldots, \mathbf{u}[t+N], \mathbf{x}[t], \ldots, \mathbf{x}[t+N]\right).$$

We denote the system state at time k by $\mathbf{x}[k]$ and the control input by $\mathbf{u}[k]$. However, without the knowledge of the dependencies of the inputs on the states, minimizing the cost function would prove useless. These dependencies are supplied in the form of the system's transition model

$$\mathbf{x}[k+1] = \mathbf{f}\left(\mathbf{x}[k], \mathbf{u}[k]\right). \tag{5.1}$$

Given the recurrent nature of (5.1), the initial state needs to be defined.

In order to further adhere to the real-world limitations, constraints on both the control inputs

$$\mathbf{u}_{min} \leq \mathbf{u}[k] \leq \mathbf{u}_{max}$$

and the states of the system

$$\mathbf{x}_{min} \leq \mathbf{x}[k] \leq \mathbf{x}_{max}$$

are introduced. Combining everything, the general optimization problem can be written as

$$\begin{aligned} \min_{\substack{\mathbf{u}[t],\ldots,\mathbf{u}[t+N-1],\\ \mathbf{x}[t],\ldots,\mathbf{x}[t+N]}} \quad & J\big(\mathbf{u}[t], \ldots, \mathbf{u}[t+N-1], \mathbf{x}[t], \ldots, \mathbf{x}[t+N]\big), \\ \text{s.t.} \quad & \mathbf{x}[k+1] = \mathbf{f}(\mathbf{x}[k], \mathbf{u}[k]), \\ & \mathbf{x}[t] = \mathbf{x}_0, \\ & \mathbf{u}_{\min} \leq \mathbf{u}[k] \leq \mathbf{u}_{\max}, \\ & \mathbf{x}_{\min} \leq \mathbf{x}[k] \leq \mathbf{x}_{\max}, \quad k = t, \ldots, t+N-1. \end{aligned} \tag{5.2}$$

Notice that since the control input at time $t+N$ influences a system state outside of the horizon, it is not included in the optimization.

5.2 MPC Control Loop

As mentioned above, the control sequence obtained by solving the optimization problem (5.2) is of an open-loop (feedforward) nature. However, since real-life systems suffer from many uncertainties such as measurement noise, environmental disturbances, etc., the open-loop approach is not suitable.

In order to create a feedback controller, the problem (5.2) is solved at each iteration with the initial state set to the current system state. The first control action from the sequence is utilized, and the rest is discarded. Based on the number N, at every iteration, a great part of the computed data is ultimately unused. The need to compute the optimization problem at every iteration also means that MPC is a computationally expensive technique.

The length of the control sequence N is often called the *prediction horizon*. The prediction horizon also specifies the amount of time into the future that the MPC predicts, which can be computed as $T = T_s N$, where T_s is the discretization period of the discrete model used.

5.2.1 Control horizon

Control horizon N_c is a term similar to the prediction horizon. The prediction horizon specifies the number of states to be optimized. The control horizon specifies the number of control actions that are to be optimized, while setting all the subsequent control actions to zero. In (5.2), the control horizon is set to be (and often it is the case) the same as the prediction horizon, but we can rewrite (5.2) with the general control horizon with the added modification

$$\begin{aligned} \mathbf{u}_{\min} \leq \mathbf{u}[j] \leq \mathbf{u}_{\max}, \quad & j = 1, \dots, N_c - 1, \\ \mathbf{u}[l] = \mathbf{0} \qquad\qquad\quad & l = N_c, \dots, N - 1. \end{aligned}$$

Introducing the control horizon lowers the number of optimization parameters, and therefore improves the computation efficiency, which can be utilized to either decrease the discretization period, thus improving the model accuracy, or increase the prediction horizon, allowing us to compare states that are further into the future.

5.3 Types of MPC

MPC is a broad field, and therefore a lot of subclasses of MPC were introduced. MPC types can be divided based either on the transition model or the formulation of the cost function. If a nonlinear model is used, we call it Nonlinear Model Predictive Control (NMPC), otherwise, using a linear model categorizes it under Linear Model Predictive Control (LMPC).

The choice of the cost function might be even more important, as improper definition might lead the optimization to converge to local minima, rather than global. The cost function can be defined even nonlinearly, but most commonly, cost functions from the regions of Linear Programming (LP) or Quadratic Programming (QP) are used. LP minimization can be formulated as

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{c}^\top \mathbf{x}, \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{b}, \\ & \mathbf{x} \in \mathbb{R}^n, \end{aligned}$$

where notation $\mathbf{Ax} \geq \mathbf{b}$ means that every i-th component of $\mathbf{Ax}$ is greater than or equal to the i-th component of the vector $\mathbf{b}$. QP assumes the form of

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b}^\top \mathbf{x}, \\ \text{s.t.} \quad & \mathbf{C}\mathbf{x} \leq \mathbf{d}, \\ & \mathbf{E}\mathbf{x} = \mathbf{f}, \\ & \mathbf{x} \in \mathbb{R}^n. \end{aligned} \tag{5.3}$$

When LP or QP are combined with a linear model, the minimization problem of the MPC iteration becomes convex[1], meaning every local minimum is simultaneously a global minimum, making the optimization much simpler. Moreover, optimizing a NMPC is much more time consuming, and with a highly agile system as an UAV, the longer computing times could introduce instability. Therefore, we chose to implement LMPC with a quadratic cost function, often called Quadratic Programming Model Predictive Control (QP-MPC), which we will cover in more detail.

5.4 Formulation of QP-MPC

Combining the use of the linear model and the quadratic cost function defined in (5.3), we can formulate the cost function for the regulation task, i.e. setting all the states to be zero, as follows:

$$\begin{aligned} \min_{\substack{\mathbf{u}[t],\ldots,\mathbf{u}[t+N-1],\\ \mathbf{x}[t],\ldots,\mathbf{x}[t+N]}} \quad & \frac{1}{2}\mathbf{x}^\top[t+N]\mathbf{S}\mathbf{x}[t+N] + \frac{1}{2}\sum_{k=t}^{t+N-1}\left(\mathbf{x}^\top[k]\mathbf{Q}\mathbf{x}[k] + \mathbf{u}^\top[k]\mathbf{R}\mathbf{u}[k]\right), \\ \text{s.t.} \quad & \mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k], \\ & \mathbf{x}[t] = \mathbf{x}_0, \\ & \mathbf{u}_{min} \leq \mathbf{u}[k] \leq \mathbf{u}_{max}, \\ & \mathbf{x}_{min} \leq \mathbf{x}[k] \leq \mathbf{x}_{max}, \quad k = t, \ldots, t+N-1. \end{aligned} \tag{5.4}$$

Leaving the individual states $\mathbf{x}[t], ..., \mathbf{x}[t+N]$ in the optimization, we get what is called the Simultaneous Formulation. We can also get rid of all the states but the initial one by using the transfer function, since every state can be obtained using the state and input at the previous iteration; this is often called the Sequential Formulation.

5.4.1 Simultaneous (Sparse) Formulation

Firstly, we stack the states and control inputs into two long vectors

$$\bar{\mathbf{x}}^\top = \begin{bmatrix} \mathbf{x}^\top[t+1] & \mathbf{x}^\top[t+2) & \ldots & \mathbf{x}^\top[t+N] \end{bmatrix}, \tag{5.5}$$

$$\bar{\mathbf{u}}^\top = \begin{bmatrix} \mathbf{u}^\top[t] & \mathbf{u}^\top[t+1] & \ldots & \mathbf{u}^\top[t+N-1] \end{bmatrix}. \tag{5.6}$$

[1] The matrix $\mathbf{A}$ in QP needs to be positively semi-definite.

We can then rewrite the cost function (5.4) as

$$\begin{aligned}\min_{\bar{\mathbf{u}},\bar{\mathbf{x}}} \quad & \frac{1}{2}\begin{bmatrix}\mathbf{x}^\top[t+1]\dots & \mathbf{x}^\top[t+N]\end{bmatrix}\underbrace{\begin{bmatrix}\mathbf{Q} & & \\ & \ddots & \\ & & \mathbf{S}\end{bmatrix}}_{\bar{\mathbf{Q}}}\begin{bmatrix}\mathbf{x}[t+1]\\ \vdots \\ \mathbf{x}[t+N]\end{bmatrix}+ \\ & \frac{1}{2}\begin{bmatrix}\mathbf{u}^\top[t]\dots & \mathbf{u}^\top[t+N-1]\end{bmatrix}\underbrace{\begin{bmatrix}\mathbf{R} & & \\ & \ddots & \\ & & \mathbf{R}\end{bmatrix}}_{\bar{\mathbf{R}}}\begin{bmatrix}\mathbf{u}[t]\\ \vdots \\ \mathbf{u}[t+N-1]\end{bmatrix}+ \\ & \underbrace{\frac{1}{2}\mathbf{x}^\top[t]\mathbf{Q}\mathbf{x}[t]}_{\text{constant}}\end{aligned} \tag{5.7}$$

subject to

$$\begin{bmatrix}\mathbf{x}[t+1]\\ \mathbf{x}[t+2]\\ \mathbf{x}[t+3]\\ \vdots \\ \mathbf{x}[t+N]\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{0} & & & & \\ \mathbf{A} & \mathbf{0} & & & \\ & \mathbf{A} & \mathbf{0} & & \\ & & & \ddots & \\ & & & \mathbf{A} & \mathbf{0}\end{bmatrix}}_{\bar{\mathbf{A}}}\begin{bmatrix}\mathbf{x}[t+1]\\ \mathbf{x}[t+2]\\ \mathbf{x}[t+3]\\ \vdots \\ \mathbf{x}[t+N]\end{bmatrix}+\underbrace{\begin{bmatrix}\mathbf{B} & & & \\ & \mathbf{B} & & \\ & & \mathbf{B} & \\ & & & \ddots \\ & & & & \mathbf{B}\end{bmatrix}}_{\bar{\mathbf{B}}}\begin{bmatrix}\mathbf{u}[t]\\ \mathbf{u}[t+1]\\ \mathbf{u}[t+2]\\ \vdots \\ \mathbf{u}[t+N-1]\end{bmatrix}+\underbrace{\begin{bmatrix}\mathbf{A}\\ \mathbf{0}\\ \mathbf{0}\\ \vdots \\ \mathbf{0}\end{bmatrix}}_{\bar{\mathbf{A}}_0}\mathbf{x}[t].$$

After stacking the vectors (5.5) and (5.6) and rewriting the terms, we arrive at the following formulation

$$\begin{aligned}\min_{\bar{\mathbf{u}},\bar{\mathbf{x}}} \quad & \frac{1}{2}\begin{bmatrix}\bar{\mathbf{x}}^\top & \bar{\mathbf{u}}^\top\end{bmatrix}\begin{bmatrix}\bar{\mathbf{Q}} & \\ & \bar{\mathbf{R}}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}\\ \bar{\mathbf{u}}\end{bmatrix}, \\ \text{s.t.} \quad & \mathbf{0} = \begin{bmatrix}(\bar{\mathbf{A}}-\mathbf{I}) & \bar{\mathbf{B}}\end{bmatrix}\begin{bmatrix}\bar{\mathbf{x}}\\ \bar{\mathbf{u}}\end{bmatrix}+\bar{\mathbf{A}}_0\mathbf{x}[t], \\ & \mathbf{x}[t] = \mathbf{x}_0, \\ & \bar{\mathbf{u}}_{min} \le \bar{\mathbf{u}} \le \bar{\mathbf{u}}_{max}, \\ & \bar{\mathbf{x}}_{min} \le \bar{\mathbf{x}} \le \bar{\mathbf{x}}_{max},\end{aligned} \tag{5.8}$$

where $\bar{\mathbf{u}}_{min}, \bar{\mathbf{u}}_{max}, \bar{\mathbf{x}}_{min}, \bar{\mathbf{x}}_{max}$ are stacked vectors of the lower and upper limits of the controls and states. Note that the last term from (5.7) has been omitted, since it is constant and therefore will not have any effect on the minimization argument.

5.4.2 Sequential (Dense) Formulation

The Sequential Formulation explores the idea that every state can be determined by the use of the transfer function and the state and control action in the previous iteration

$$\mathbf{x}[k+1] = \mathbf{A}\mathbf{x}[k] + \mathbf{B}\mathbf{u}[k].$$

Taking this idea further, we can derive the state at time [k+2] by substituting for $\mathbf{x}[k+1]$

$$\mathbf{x}[k+2] = \mathbf{A}\mathbf{x}[k+1] + \mathbf{B}\mathbf{u}[k+1] = \mathbf{A}^2\mathbf{x}[k] + \mathbf{A}\mathbf{B}\mathbf{u}[k] + \mathbf{B}\mathbf{u}[k+1]$$

and, subsequently, we can utilize this formula for an arbitrary state at any given time, where the value of the state is a function of the sequence of control inputs and the initial state $\mathbf{x}[k]$

$$\mathbf{x}[k+n] = \mathbf{A}^n\mathbf{x}[k] + \mathbf{A}^{n-1}\mathbf{B}\mathbf{u}[k] + \mathbf{A}^{n-2}\mathbf{B}\mathbf{u}[k+1] + \cdots + \mathbf{B}\mathbf{u}[k+n-1].$$

Rewriting this into matrix form notation, we get

$$\begin{bmatrix} \mathbf{x}[t+1] \\ \mathbf{x}[t+2] \\ \vdots \\ \mathbf{x}[t+N] \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{B} & & & \\ \mathbf{AB} & \mathbf{B} & & \\ \vdots & & \ddots & \\ \mathbf{A}^{N-1}\mathbf{B} & \mathbf{A}^{N-2}\mathbf{B} & & \mathbf{B} \end{bmatrix}}_{\hat{\mathbf{C}}} \begin{bmatrix} \mathbf{u}[t] \\ \mathbf{u}[t+1] \\ \vdots \\ \mathbf{u}[t+N-1] \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{A} \\ \mathbf{A}^2 \\ \vdots \\ \mathbf{A}^N \end{bmatrix}}_{\hat{\mathbf{A}}}.$$

Using the notation with the stacked vectors (5.5), (5.6), the whole transition constraint can be written as

$$\bar{\mathbf{x}} = \hat{\mathbf{C}}\bar{\mathbf{u}} + \hat{\mathbf{A}}\mathbf{x}[t].$$

Substituting into the cost function (5.8), we get a cost function that becomes independent of $\bar{\mathbf{x}}$

$$\begin{aligned} \tilde{J}(\bar{\mathbf{u}}, \mathbf{x}[t]) &= \frac{1}{2}(\hat{\mathbf{C}}\bar{\mathbf{u}} + \hat{\mathbf{A}}\mathbf{x}[t])^\top \bar{\mathbf{Q}}(\hat{\mathbf{C}}\bar{\mathbf{u}} + \hat{\mathbf{A}}\mathbf{x}[t]) + \frac{1}{2}\bar{\mathbf{u}}^\top\bar{\mathbf{R}}\bar{\mathbf{u}} + \frac{1}{2}\mathbf{x}^\top[t]\mathbf{Q}\mathbf{x}[t] \\ &= \frac{1}{2}\bar{\mathbf{u}}^\top\hat{\mathbf{C}}^\top\bar{\mathbf{Q}}\hat{\mathbf{C}}\bar{\mathbf{u}} + \mathbf{x}^\top[t]\hat{\mathbf{A}}^\top\bar{\mathbf{Q}}\hat{\mathbf{C}}\bar{\mathbf{u}} + \frac{1}{2}\mathbf{x}^\top[t]\hat{\mathbf{A}}^\top\bar{\mathbf{Q}}\hat{\mathbf{A}}\mathbf{x}[t] + \frac{1}{2}\bar{\mathbf{u}}^\top\bar{\mathbf{R}}\bar{\mathbf{u}} + \frac{1}{2}\mathbf{x}^\top[t]\mathbf{Q}\mathbf{x}[t] \\ &= \frac{1}{2}\bar{\mathbf{u}}^\top \underbrace{(\hat{\mathbf{C}}^\top\bar{\mathbf{Q}}\hat{\mathbf{C}} + \bar{\mathbf{R}})}_{\mathbf{H}}\bar{\mathbf{u}} + \mathbf{x}^\top[t]\underbrace{\hat{\mathbf{A}}^\top\bar{\mathbf{Q}}\hat{\mathbf{C}}}_{\mathbf{F}^\top}\bar{\mathbf{u}} + \underbrace{\frac{1}{2}\mathbf{x}^\top[t](\hat{\mathbf{A}}^\top\bar{\mathbf{Q}}\hat{\mathbf{A}} + \mathbf{Q})\mathbf{x}[t]}_{\text{constant}}. \end{aligned}$$

The last term can once again be omitted since the optimizer will not be affected by it. Using all of the above, the optimization problem takes the form

$$\begin{aligned} \min_{\bar{\mathbf{u}}} \quad & \frac{1}{2}\bar{\mathbf{u}}^\top\mathbf{H}\bar{\mathbf{u}} + \mathbf{x}^\top[t]\mathbf{F}^\top\bar{\mathbf{u}}, \\ \text{s.t.} \quad & \mathbf{x}[t] = \mathbf{x}_0, \\ & \bar{\mathbf{u}}_{min} \le \mathbf{u}[k] \le \bar{\mathbf{u}}_{max}, \\ & \bar{\mathbf{x}}_{min} \le \hat{\mathbf{C}}\bar{\mathbf{u}} + \hat{\mathbf{A}}\mathbf{x}[t] \le \bar{\mathbf{x}}_{max}. \end{aligned}$$

Most often than not, solving the MPC optimization problem requires an iterative solver. However, omitting the constraints on states and control actions, we can arrive at a closed-form solution, by taking the gradient of the cost function and setting it to zero

$$\nabla_{\bar{\mathbf{u}}}\tilde{J}(\bar{\mathbf{u}}, \mathbf{x}[t]) = \mathbf{H}\bar{\mathbf{u}} + \mathbf{F}\mathbf{x}[t] = \mathbf{0},$$

we can find the local extreme of the cost function. Since the problem is convex, setting the gradient to zero effectively finds the global minimum, and upon solving, the solution can be written as

$$\bar{\mathbf{u}} = -\mathbf{H}^{-1}\mathbf{F}\mathbf{x}[t].$$

Since MPC utilizes only the first control action from the sequence, the solution can be adjusted to return the desired control action directly

$$\mathbf{u}[t] = \underbrace{-\begin{bmatrix}\mathbf{I} & \mathbf{0} & \dots & \mathbf{0}\end{bmatrix}\mathbf{H}^{-1}\mathbf{F}}_{\mathbf{K}}\mathbf{x}[t].$$

Combining the matrices, we can notice that the immediate control action is obtained as $\mathbf{u}[t] = \mathbf{K}\mathbf{x}[t]$, therefore, QP-MPC without constraint acts as a special case of state feedback.

5.4.3 Partial Condensing

A question arises as to which formulation should be used. The dense approach might seem like the obvious choice due to the elimination of variables, which should make it more efficient. This is especially true for small control horizons. However, as the prediction horizon grows, the matrices can become nearly singular. Sparse problems, on the other hand, produce near-diagonal matrices with most elements being zero. Solvers can exploit this sparsity to reduce both the memory footprint and the computation time of the optimization.

Partial condensing [33] lies between these two approaches. The basic idea is to divide the prediction horizon into M blocks, eliminating the state variables within each one using condensation (formulating the problem as dense). This results in a matrix composed of diagonally connected dense sub-matrices corresponding to the individual blocks. By adjusting the parameter M, we can gradually pass from the sparse to the dense formulation, without the need to strictly choose one over the other.

5.4.4 Reference Tracking

So far, we have focused on solving the regulation task, which involves bringing all the states to zero. To reformulate the problem for reference tracking, the objective function needs to be modified to minimize the tracking error $\mathbf{e}[t]$ instead of the states. The tracking error is defined as the difference between the reference state $\mathbf{x}_r[t]$ and the actual state[2] $\mathbf{x}[t]$, i.e. $\mathbf{e}[t] = \mathbf{x}_r[t] - \mathbf{x}[t]$. Substituting the tracking error into the cost function (5.4) yields

$$\begin{aligned} J &= \frac{1}{2}\sum_{k=t}^{t+N-1}\Big((\mathbf{x}_r[k]-\mathbf{x}[k])^\top\mathbf{Q}(\mathbf{x}_r[k]-\mathbf{x}[k]) + \mathbf{u}^\top[k]\mathbf{R}\mathbf{u}[k]\Big) \\ &\quad + \frac{1}{2}(\mathbf{x}_r[t+N]-\mathbf{x}[t+N])^\top\mathbf{S}(\mathbf{x}_r[t+N]-\mathbf{x}[t+N]) \\ &= \sum_{k=t}^{t+N-1}\left(\underbrace{\frac{1}{2}\mathbf{x}_r^\top[k]\mathbf{Q}\mathbf{x}_r[k]}_{\text{constant}} - \mathbf{x}^\top[k]\mathbf{Q}\mathbf{x}_r[k] + \frac{1}{2}\mathbf{x}^\top[k]\mathbf{Q}\mathbf{x}[k] + \mathbf{u}^\top[k]\mathbf{R}\mathbf{u}[k]\right) \\ &\quad + \underbrace{\frac{1}{2}\mathbf{x}_r^\top[t+N]\mathbf{S}\mathbf{x}_r[t+N]}_{\text{constant}} - \mathbf{x}^\top[t+N]\mathbf{S}\mathbf{x}_r[t+N] + \frac{1}{2}\mathbf{x}^\top[t+N]\mathbf{S}\mathbf{x}[t+N]. \end{aligned} \tag{5.9}$$

Note that the constant terms can once again be omitted from the optimization.

[2] The tracking error can also be defined as the difference between a reference output and the system's current output, with the use of the state to output matrix $\mathbf{C}$.

6 Implementation

Building upon Chapters 3, 4 and 5, this chapter describes the practical implementation of the entire control pipeline. All algorithms were written in C++ within the Robot Operating System (ROS) framework, and inside a Multi-robot Systems Group (MRS) workspace. The algorithm is intended to run on the UAV's onboard computer, such as the Intel NUC. Development, debugging, and tuning took place in the MRS system simulator [34], the backbone of which is the physics simulator Gazebo [35]. This allowed for near-real-life behavior of the systems, making the later deployment in the real world much easier.

Since the measurement approaches for the leader's position and attitude are unavailable, we obtain these measurements through the leader's onboard estimator and transmit them via Wi-Fi to the follower UAV. To simulate measurement noise, we introduce it artificially during each iteration of the controller.

6.1 Kalman Filter

The LKF was implemented as a standalone class in the workspace, with both the prediction and correction steps being callable methods. Matrix operations are handled using the *Eigen* library, ensuring efficient computation and ease of use. The only parameter required by the LKF model is the leader's mass, which is assumed to be known. In simulations, the equal mass to the follower's was used, while in real-world experiments, the leader UAV was physically weighed.

The prediction method accepts the time between consecutive controller iterations and uses it to discretize the model (3.4) via the Forward Euler method, as described in section 2.8. This dynamic discretization is crucial, as the controller's iteration period is not guaranteed to be constant, despite being designed to run at 100 Hz. The method then performs the prediction step (3.1), storing the resulting state and covariance estimates internally.

The correction method takes the current noisy measurements of the leader's position and attitude, applies the correction step (3.2) and updates the internally saved estimates accordingly. We chose to leave the yaw (and the corresponding angular velocity) in the LKF model for completeness, and supply zero at all times as the yaw measurement.

To evaluate the effects of the leader's attitude measurements on the overall performance, a second variant of the LKF was implemented, which utilizes only the measurements of the position. The implementation remains largely identical, with the only difference being a cropped system output matrix (3.5)[1].

6.1.1 Acceleration Bias

In order to further adhere to real-world behavior, we introduce two additional states: an acceleration bias in the x and y direction. These states are designed to correct the acceleration estimation of the model, which could be caused by sensor bias, real-world disturbances, or an inaccurate model. We therefore adjust the velocity differential equations from (3.3) as

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \begin{bmatrix} \frac{T}{m} (\cos\psi \sin\theta \cos\phi + \sin\psi \sin\phi) - b_x \\ \frac{T}{m} (\sin\psi \sin\theta \cos\phi - \cos\psi \sin\phi) - b_y \end{bmatrix},$$

[1] The rows for the attitude outputs are omitted from the matrix.

where b_x, b_y are the acceleration biases for the x and y axis, respectively. The differential equations for the offset states are unknown; we therefore once again model them as constant, setting their differential equations to zero. The linear system state matrix (3.4) is updated accordingly.

6.1.2 Tuning the LKF

Tuning the LKF parameters consists of setting the diagonal values of the $\mathbf{Q}_{KF}$ and $\mathbf{R}_{KF}$ matrices. The filter's behavior is determined by these values. The measurement noise covariance matrix $\mathbf{R}_{KF}$ reflects the confidence in the sensor measurements, while the process noise covariance matrix $\mathbf{Q}_{KF}$ reflects the confidence in the model dynamics. Setting the $\mathbf{R}_{KF}$ values too low causes the filter to overly trust the noisy measurements, while setting them too high makes the filter ignore the measurements entirely. If the $\mathbf{Q}_{KF}$ values are too low, the filter puts too much confidence in the dynamic model, reacting too slowly to changes. Setting the values too high removes the confidence in the model, resulting in an overly responsive performance and noisy estimations. The filter's behavior inherently depends on the relative values of both matrices rather than the absolute ones.

Since the $\mathbf{R}_{KF}$ matrix ideally reflects the actual sensor noise, its values were derived from real sensor characteristics. For the position measurements, we assume the use of Real-Time Kinematic Global Positioning System (RTK-GPS), allowing for centimeter accuracy. Assuming the accuracy of 5 cm, this gives the standard deviation of position $\sigma_P = 0.05\,\text{m}$, and thus a covariance $\sigma_P^2 = 0.0025\,\text{m}$.

For attitude, a covariance matrix

$$\Sigma_{\hat{\mathbf{z}}} = \begin{bmatrix} 0.0277 & -0.00279 & -0.0041 \\ -0.00279 & 0.00937 & 0.00002 \\ -0.0041 & 0.00002 & 0.000379 \end{bmatrix}$$

was obtained from a ML-based method, which estimates the orientation of the z-axis of the Body frame $\hat{\mathbf{z}} = \begin{bmatrix} x_z & y_z & z_z \end{bmatrix}^\top$. As the heading (yaw angle) is assumed to be always zero, the z-axis orientation is enough to obtain the roll and pitch angles. We can transform the covariance matrix of the z-axis estimation into a covariance matrix of the roll and pitch angles by using a linear transformation. Using the approximation $\alpha \approx \sin\alpha$ for small angles, the transformation from the z-axis to the angles can be approximated as

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} \approx \underbrace{\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{T} \begin{bmatrix} x_z \\ y_z \\ z_z \end{bmatrix}.$$

The covariance matrix can then be obtained by using the formula $\Sigma_A = T\Sigma_{\hat{\mathbf{z}}}T^\top$, where T is a linear transformation matrix. Substituting into the equation yields

$$\Sigma_A = \begin{bmatrix} 0.0094 & -0.00279 \\ -0.00279 & 0.0278 \end{bmatrix}.$$

Taking a larger covariance and accounting for the linearization error, we set the covariance for the attitude as $\sigma_A^2 = 0.03\,\text{rad}^2$.

$\mathbf{Q}_{KF}$ was tuned empirically through trial and error in the simulation, where the noise for both attitude and position was added artificially in each iteration of the controller. Table 6.1 shows the values of the final $\mathbf{Q}_{KF}$ and $\mathbf{R}_{KF}$ matrices.

$\mathbf{Q}_{LKF}$	diag([	x	y	z	v_x	v_y	v_z	ϕ	θ	ψ	ω_ϕ	ω_θ	ω_ψ	T	o_x	o_y	])
		e^{-3}			$6e^{-3}$	e^{-3}		0			$8e^{-4}$			0.1	e^{-4}		
$\mathbf{R}_{LKF}$	diag([	0.0025	0.0025	0.0025	0.03	0.03	0.03	])									
		x	y	z	ϕ	θ	ψ										

Table 6.1: The tuned process noise covariance and measurement noise covariance matrices. The empty places in the $\mathbf{Q}_{LKF}$ denote the same number as the previous one in order to save space.

6.2 Trajectory Prediction

The trajectory prediction is implemented as a method inside the LKF class and follows the pseudo-algorithm described in Algorithm 1. The linear predictions are replaced with the nonlinear ones with the use of equations (3.3). The prediction's starting point is the current state estimation obtained from the LKF. The predictions are computed twice: once with the estimation from the LKF with attitude measurements and once using only the position-based estimate[2]. The number of predicted steps and the sampling period match the prediction horizon N and the discretization period of the MPC, enabling seamless use of the predicted trajectory as the reference.

6.3 MPC

The implementation of the MPC controller follows the concept and structure described in chapter 5. The controller has been implemented with the use of the *Acados* [36] framework. Acados is an open-source framework that unifies the interface for state-of-the-art solvers for the MPC optimization problem, like High-Precision Interior Point Method (HPIPM), which was used in this thesis.

6.3.1 Problem Formulation

The HPIPM solver formulates the cost function in the following way:

$$\min_{\bar{\mathbf{x}},\bar{\mathbf{u}}} \frac{1}{2} \sum_{n=0}^{N} \begin{bmatrix} \mathbf{u}_n \\ \mathbf{x}_n \\ 1 \end{bmatrix}^\top \begin{bmatrix} \mathbf{R}_n & \mathbf{S}_n & \mathbf{r}_n \\ \mathbf{S}_n & \mathbf{Q}_n & \mathbf{q}_n \\ \mathbf{r}_n & \mathbf{q}_n & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_n \\ \mathbf{x}_n \\ 1 \end{bmatrix}.$$

Multiplying the terms and comparing them to the derived reference cost function in (5.9), we arrive at the following substitutions

$$\mathbf{r}_n = \mathbf{0},$$
$$\mathbf{S}_n = \mathbf{0},$$
$$\mathbf{q}_n = -\mathbf{Q}_n \mathbf{x}_n^r,$$

where $\mathbf{x}_n^r$ is the reference state. We can therefore see that setting the reference is done by setting the linear term $\mathbf{q}_n$. The $\mathbf{R}_n$ and $\mathbf{Q}_n$ represent the quadratic costs in relation to the input and state, respectively. The subscript n tells us that these matrices can be time-dependent and changed for every step of the MPC.

[2] This is done in order to compare the two approaches.

6.3.2 Incremental Model

In order to have more precise control over the behavior of the control inputs outputted by the MPC, we transform our model into an incremental (delta-u) one. Incremental models make the control actions part of the states, concatenating the input vector to the state vector. The input vector is replaced by a vector of input increments. We can transform the LTI discretized model to an incremental one as follows:

$$\begin{bmatrix}\mathbf{x}[k+1]\\ \mathbf{u}[k+1]\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}}_{\mathbf{A}_{INC}} \begin{bmatrix}\mathbf{x}[k]\\ \mathbf{u}[k]\end{bmatrix} + \underbrace{\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}}_{\mathbf{B}_{INC}} \Delta\mathbf{u}[k].$$

Making this change, the states of our model become

$$\mathbf{x}_{INC} = \begin{bmatrix} x & y & z & v_x & v_y & v_z & \phi & \theta & \psi & T & \phi_R & \theta_R & T_R \end{bmatrix},$$

while the inputs take the form of

$$\mathbf{u}_{INC} = \begin{bmatrix}\Delta\phi_R & \Delta\theta_R & \Delta T_R\end{bmatrix}.$$

Changing the model to incremental allows us to regulate the input increments through the $\mathbf{R}_n$ matrix, thus making the change in the inputs more conservative and therefore the control actions much smoother.

6.3.3 Setting the Reference

After obtaining the predicted states, we can set the reference for every timestep n with the use of the linear term $\mathbf{q}_n = -\mathbf{Q}\mathbf{x}_n^r$. However, using the same cost matrix $\mathbf{Q}$ for every step would mean that we put the same emphasis on every reference. This is a flawed approach, as the positions at the end of the predicted trajectory are less likely to be achieved by the leader UAV, while the positions at the start of the trajectory are much more probable. This is due to the disturbances in the dynamics or the inaccurate assumption that the angular velocities and thrust in the LKF model stay constant.

We therefore use the covariance matrix to scale the cost matrix $\mathbf{Q}$, resulting in a different cost matrix $\mathbf{Q}_n$ for every step n. The bigger the values in the covariance matrix, the less confidence we have in the corresponding reference. By calculating the pseudo-inverse of the covariance matrix, we receive a near-diagonal matrix. The values on the diagonals can be used to scale the corresponding values in the $\mathbf{Q}$ matrix. By utilizing this approach, we effectively make the references with bigger covariance less influential.

Through experiments, we have found that the scaling by the covariance is not enough, as the MPC still puts too much trust in the references at the end of the prediction, making the follower UAV overtake the leader on linear trajectories. To compensate, we scaled the individual cost matrices $\mathbf{Q}_n$ at every timestep n based on the elapsed time as

$$\mathbf{Q}_n \leftarrow e^{-\alpha n T_s}\mathbf{Q}_n,$$

where T_s is the discretization period of the predictions. The parameter α was iteratively chosen as $\alpha = 2$.

Outside of the reference, the starting state $\mathbf{x}_0$ of the UAV also needs to be initialized. We set the state as the current state estimation, provided by an onboard estimator, together with the last control action, as we are using an incremental model.

6.3.4 Constraints

We want to introduce constraints on the states for 2 reasons; firstly, deviating far apart from the equilibrium point makes the linear approximations less accurate. Secondly, we want to avoid dangerous situations, which could result in a loss of control, or damage to the aircraft, f. e. turning the UAV upside down. For these reasons, we set constraints on the roll ϕ and pitch θ angles, and vertical velocity of the aircraft. The specific values are shown in Table 6.2. The rest of the states we leave unconstrained, as there is no reason to do otherwise.

value	min	max
ϕ, θ	$-0.5\,\mathrm{rad}$	$0.5\,\mathrm{rad}$
v_x, v_y	$-10\,\mathrm{ms}^{-1}$	$10\,\mathrm{ms}^{-1}$

Table 6.2: Introduced constraints on the UAV states.

6.3.5 Tuning the MPC

The MPC controller can be tuned to achieve the desired behavior by changing numerous parameters. These parameters include

(i) **The general solver parameters**: The prediction/control horizon, the sampling period, etc.
(ii) **The cost function**: Setting the diagonal values in $\mathbf{Q}_n$ and $\mathbf{R}_n$ matrices.

We set the prediction horizon N as big as the hardware allows us to, i. e. when the time taken to solve the optimization does not pose a problem. Based on how far into the future we want to predict, the sampling period T_s is chosen and can be calculated as $t_{\max} = NT_s$. The whole set of parameters is shown in Table 6.3.

	N	N_C	T_s	Formulation	M
Value:	50	50	0.05	Partial condensing	10

Table 6.3: Overview of the tuned MPC parameters, N is the prediction horizon, N_C is the control horizon, T_s is the sampling period, and M is the level of condensation.

The cost function was tuned using the simulation environment in order to satisfy the following requirements:

(i) Fast and accurate position tracking;
(ii) Smooth control input change, i. e. no rapid changes back and forth;
(iii) No overly aggressive inputs.

Table 6.4 shows the overview of the cost matrices for the states and control inputs. Note that the final matrix $\mathbf{Q}_n$ for timestep n is calculated as the scaled version of the matrix $\mathbf{Q}$, as described in subsection 6.3.3.

		x	y	z	v_x	v_y	v_z	ϕ	θ	ψ	T	ϕ_R	θ_R	T_R	
Q	diag([	100	100	500	5	5	5	0	0	0	0	2000	2000	10	])
		$\Delta\phi_R$		$\Delta\phi_R$		$\Delta\phi_R$									
R	diag([	100000		100000		1000	])								

Table 6.4: The tuned MPC cost matrices **Q** and **R**.

7 Experiments

This chapter presents the results of the conducted experiments, evaluating the performance of all the individual components of the algorithm, as well as the overall effectiveness of the pipeline. Firstly, the simulation results are presented, containing a thorough evaluation of the algorithm's efficiency and the effect of the attitude measurements, plus the impact of different noise levels and refresh rates of the measurements. In the end, we present the results of real-life experiments, showing the algorithm's usability outside of the simulations.

In all simulation experiments, the refresh rate of the controller was set to 50 Hz. Normally, the established rate is 100 Hz, but due to hardware limitations, we had to slow down the simulation by a factor of five in order to achieve even the 50 Hz controller rate. Since the time required for the experiments was already increased fivefold, we settled on 50 Hz, as it provided satisfactory results. Unless stated otherwise, the following parameters were used for the experiments:

- attitude measurement refresh rate ... 50 Hz,
- attitude noise covariance σ_A^2 ... $0.03\,\mathrm{rad}^2$,
- position measurement refresh rate ... 50 Hz,
- position noise covariance σ_A^2 ... $0.0025\,\mathrm{m}^2$.

For the LKF and MPC, the parameters described in chapter 6 were utilized.

The experiments were conducted using a line trajectory for the leader UAV. The line trajectory consists of the leader moving 10 meters in the y-axis direction and back to its starting point, repeating. The trajectory was flown at a constant height of 5 meters. The speed of the trajectory was given by a time parameter dt, which specified the time of travel between the individual setpoints. For example, the line trajectory is given by setpoints $[0, 0, 5]$(x,y,z) and $[0, 10, 5]$, and setting the $dt = 3\,\mathrm{s}$ means that the leader has 3 seconds to move from one point to the other. Notably, when the dt is set too low, the leader does not have enough time to cover the specified distance, meaning the turn on the line begins earlier. The experiments were made for 3 separate leader speeds given by $dt = 2/3/4$ s. This allows us to compare the performance of the controllers from slower up to very agile trajectories. It must be noted that setting $dt = 2\,\mathrm{s}$ does not mean that the leader is necessarily faster than with $dt = 3\,\mathrm{s}$. Since there is less time, the leader cannot achieve as high a velocity; however, the change in direction will be much faster during $dt = 2\,\mathrm{s}$.

The line trajectory was chosen specifically for several reasons. First, since the value in the x-axis remains constant, we focus solely on the performance along the y-axis. We assume that the controller behaves similarly in both the x and y axes, which was confirmed during the experiments. More importantly, this setup simplifies the interpretation of the data collected during the experiments. For the same reasons, we do not evaluate the performance along the z-axis, as the attitude measurements have a negligible effect on its control. Lastly, the line trajectory features a highly agile movement—a complete change in the direction of flight—making it ideal for our evaluation.

7.1 LKF

In this section, we evaluate and compare the performance of the LKF with and without attitude measurements—hereafter referred to as LKF$_A$(utilizes attitude measurements)

and LKF$_P$(utilizes only position measurements), respectively. The behavior of both filters is compared to the ground truth obtained from the simulation. The measurements of the leader's position and attitude were obtained from the leader's onboard estimator. Noise with the covariance specified in subsection 6.1.2 was artificially added to the measurements in each iteration to simulate the real-world behavior of the sensor readings.

Starting with the slowest motion for $dt = 4\,\mathrm{s}$ in Figure 7.1, the difference between the two filters is relatively small. We observe that both filters perform similarly in estimating the position and velocity. The LKF$_A$ estimates both the angle and angular velocity slightly better, and therefore provides a better estimation of acceleration. Interestingly, the filter without the attitude measurements does not make use of the acceleration bias at all, which remains nearly zero at all times.

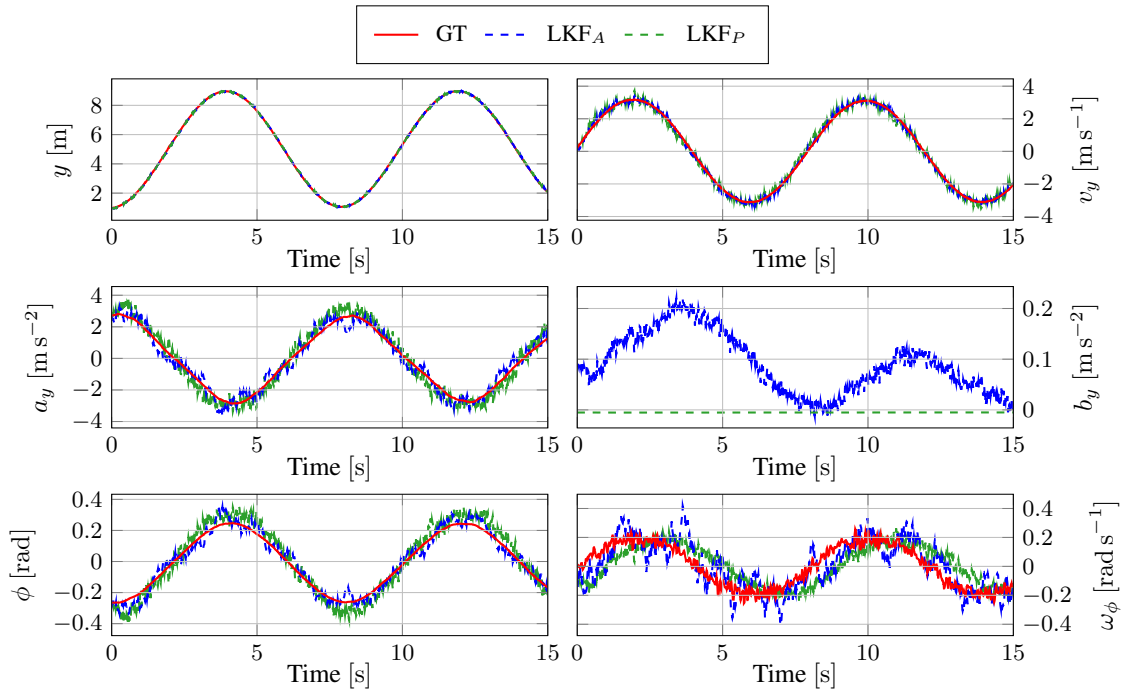


Figure 7.1: Comparison of LKF with and without attitude measurements for $dt = 4\,\mathrm{s}$.

Similar results can be concluded from the experiment with $dt = 3\,\mathrm{s}$, shown in Figure 7.2. The performance in estimating position and velocity stays relatively similar. However, as the motion becomes faster and the direction changes more abruptly, the difference in acceleration estimation becomes more significant. While LKF$_A$ provides tight tracking of acceleration thanks to the attitude measurements, the estimates of acceleration from LKF$_P$ become increasingly delayed and less accurate.

At $dt = 2\,\mathrm{s}$ shown in Figure 7.3, LKF$_P$ begins to lose accuracy even in the velocity estimates. The acceleration estimates show a consistent delay, approximately 0.3 s. This highlights the influence of the attitude measurements on the acceleration estimation, since acceleration is directly (and almost linearly) linked to the rotation, as illustrated in the roll and acceleration plots.

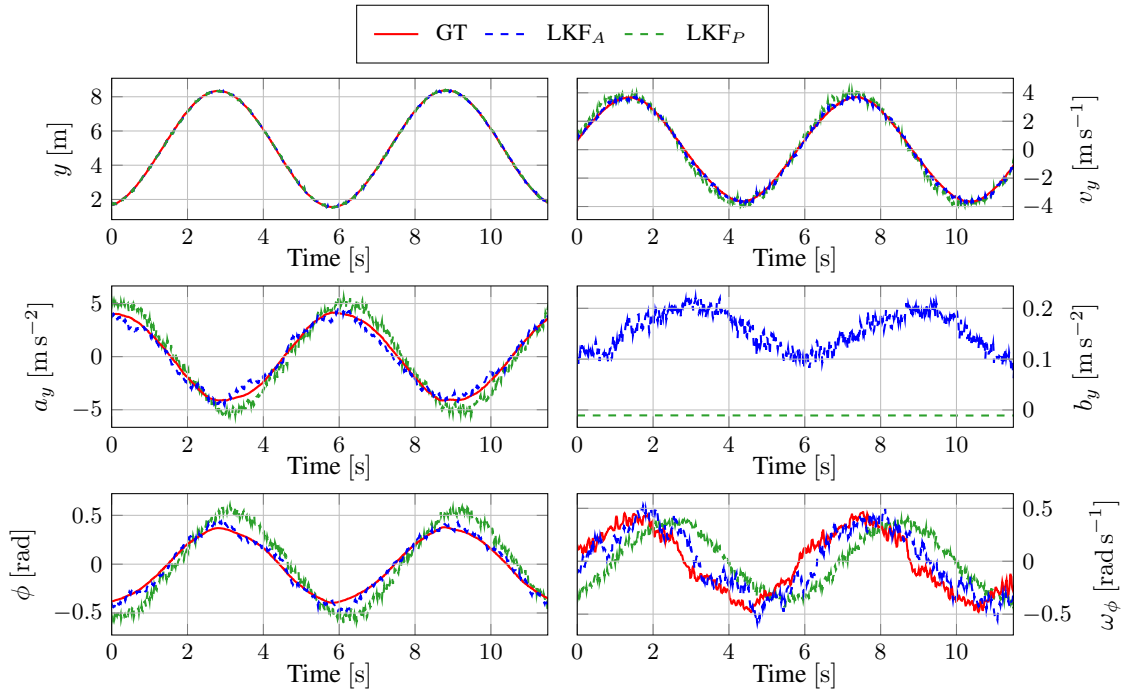


Figure 7.2: Comparison of LKF with and without attitude measurements for $dt = 3\,\mathrm{s}$.

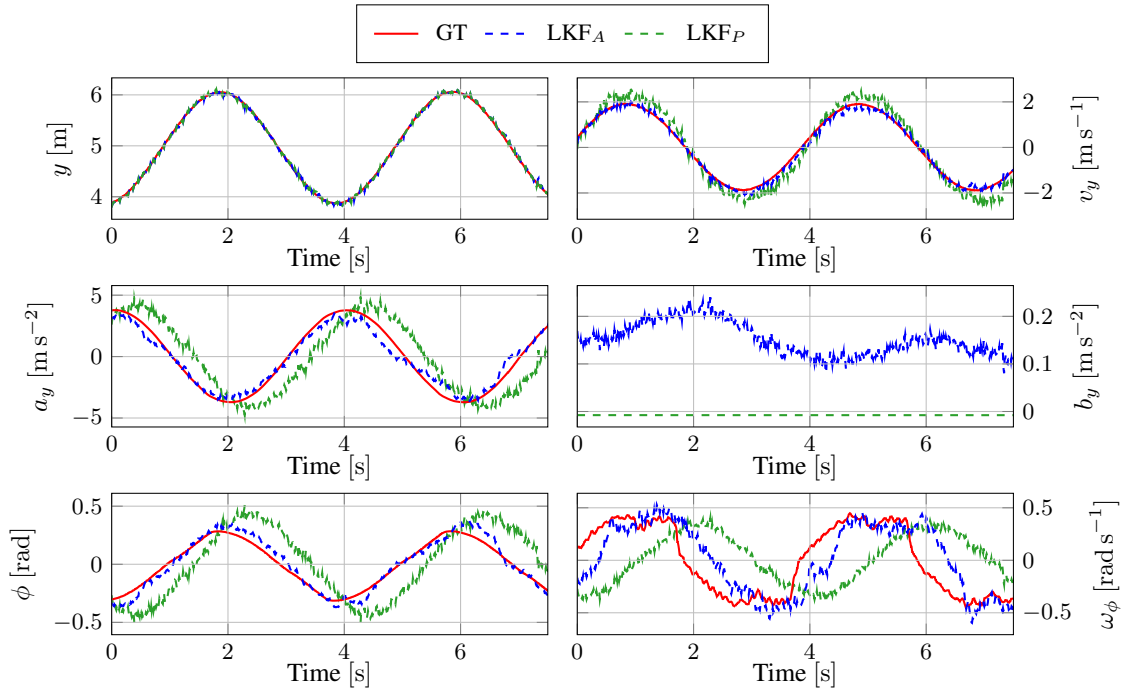


Figure 7.3: Comparison of LKF with and without attitude measurements for $dt = 2\,\mathrm{s}$.

7.2 Trajectory Predictor

We present the results of trajectory prediction solely for the case $dt = 2\,\mathrm{s}$, as the prediction performance relies only on the state estimates by the Kalman Filters, which were thoroughly evaluated in the previous section. Therefore, we focus only on the most agile trajectory tested.

In Figure 7.4, the MSE of the position prediction is depicted, with the leader's velocity and acceleration plotted for reference. The MSE for one prediction instance was calculated as follows: for every predicted state, we computed the error as the absolute difference from the predicted leader's y-position and the actual leader's y-position at the time given by the prediction. Finally, the MSE was computed from the collected error values.

As illustrated, the MSE of the predictions based on the LKF_A filter (which utilizes the attitude measurements) is approximately four times lower at the peaks compared to those based on LKF_P. An intriguing observation is that the error peaks occur at different times. For LKF_A, the largest prediction error arises at a time when the acceleration begins to change—specifically, when the leader UAV begins braking to reverse direction. In contrast, the largest deviation in LKF_P's prediction occurs when the velocity reaches its maximum[1]. Since the LKF_P estimates the acceleration poorly, it does not anticipate the braking behavior of the leader; therefore, the error gets so high.

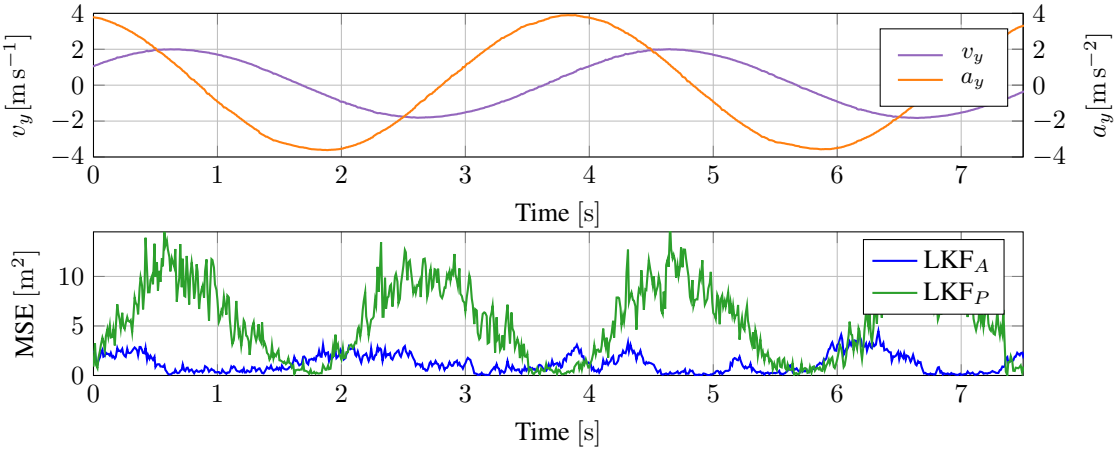


Figure 7.4: MSE of position predictions of both Kalman Filters for $dt = 2\,\mathrm{s}$. The top graph shows the leader's velocity and acceleration for context.

Figure 7.5 presents a single prediction instance, demonstrating the exact behavior described above. The prediction was made during a braking maneuver. While the prediction made by LKF_A closely tracks the leader's trajectory, LKF_P's prediction deviates significantly due to the inaccurate estimate of the acceleration.

7.3 Controller

In the following section, the performance of the entire pipeline is evaluated. We begin by comparing the controller that utilizes the attitude measurements (denoted as MPC_A) to the one that uses only position measurements (denoted as MPC_P). Following this, we analyze

[1]The term maximum is used loosely and refers to the peaks in both the positive and negative values.

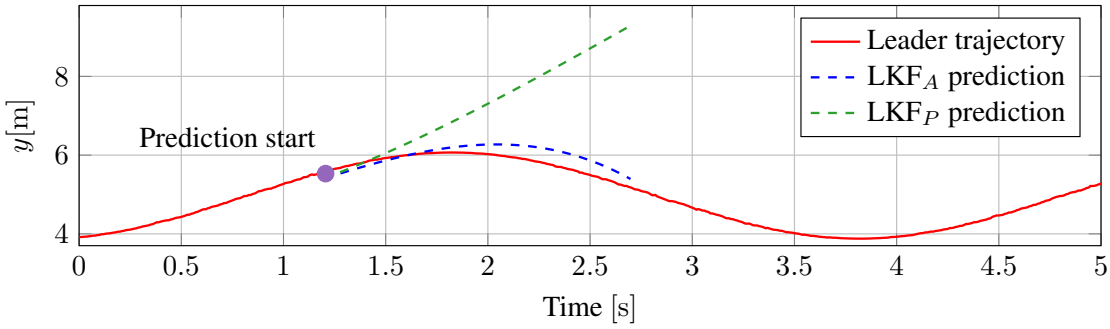


Figure 7.5: A single prediction instance of the Kalman Filters, showing the leader's position in the y-axis.

the impact of measurement noise and the update rate of the attitude measurements on the overall performance.

Although the experiments were conducted in simulation, the reference trajectory could not be reproduced exactly across all runs. The trajectories therefore occasionally differ by up to $\approx 5\,\text{cm}$ along each axis. These discrepancies are disregarded in favor of maintaining clear and concise plots.

7.3.1 Effect of Attitude Measurements

In Figure 7.6, we show the behavior of both controllers for the slowest case given by $dt = 4\,\text{s}$. Notably, the difference between the performances is not significant, with the MPC$_P$ exhibiting only a slight delay in tracking[2]. The leader and follower UAVs show similar velocity, acceleration, and roll profiles under both controllers.

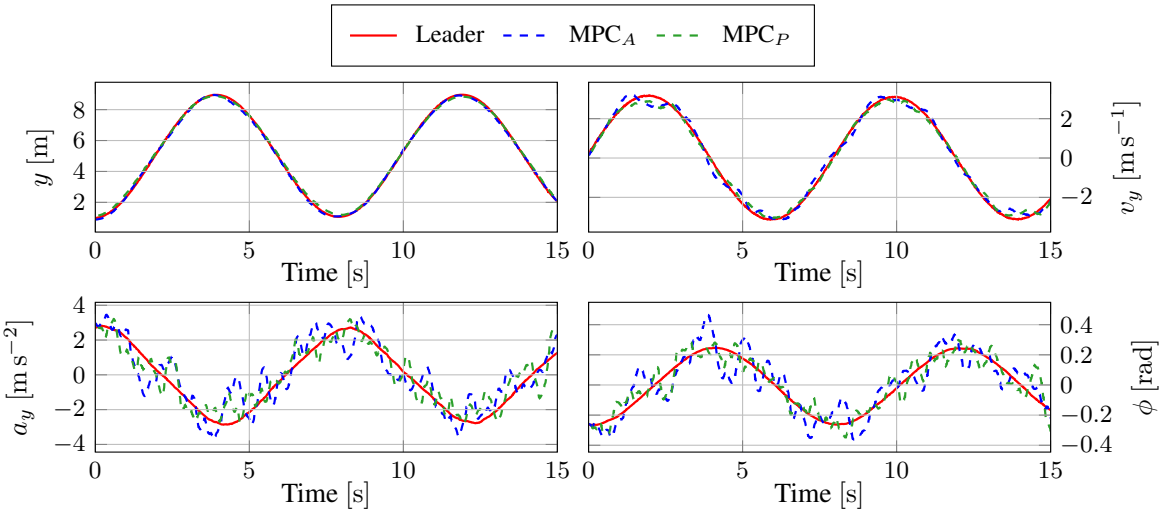


Figure 7.6: Comparison of both controllers for $dt = 4\,\text{s}$.

As the trajectory speed increases, shown in Figure 7.7, the position tracking of MPC$_A$ proves much more accurate, with the MPC$_P$ even sometimes overtaking the leader UAV, as it reacts to the braking behavior much later. In the case of $dt = 2\,\text{s}$ illustrated in Figure 7.8,

[2]The delay cannot be seen in the plot, but is apparent from the statistical results.

MPC_P's tracking error increases considerably due to the leader's agile motion. Moreover, the MPC_P compensates for its flawed prediction during the turns with more aggressive maneuvers, which are visible in the velocity, acceleration, and roll plots. In fact, the roll reaches the saturation imposed by the MPC constraints. In contrast, MPC_A not only provides accurate position tracking, but also manages to keep similar velocity and acceleration as the leader throughout the trajectory.

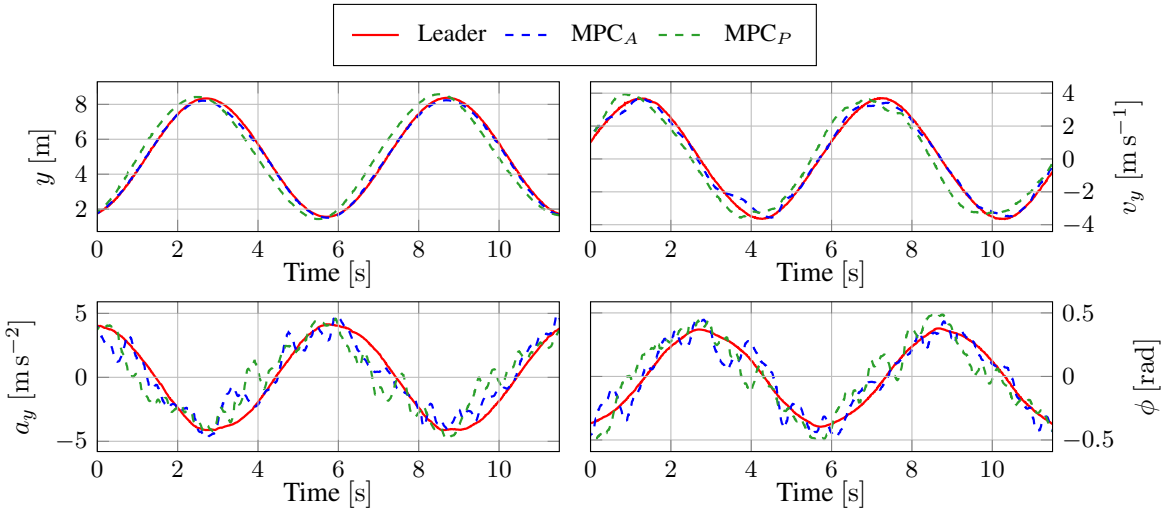


Figure 7.7: The comparison of both controllers for $dt = 3\,\text{s}$.

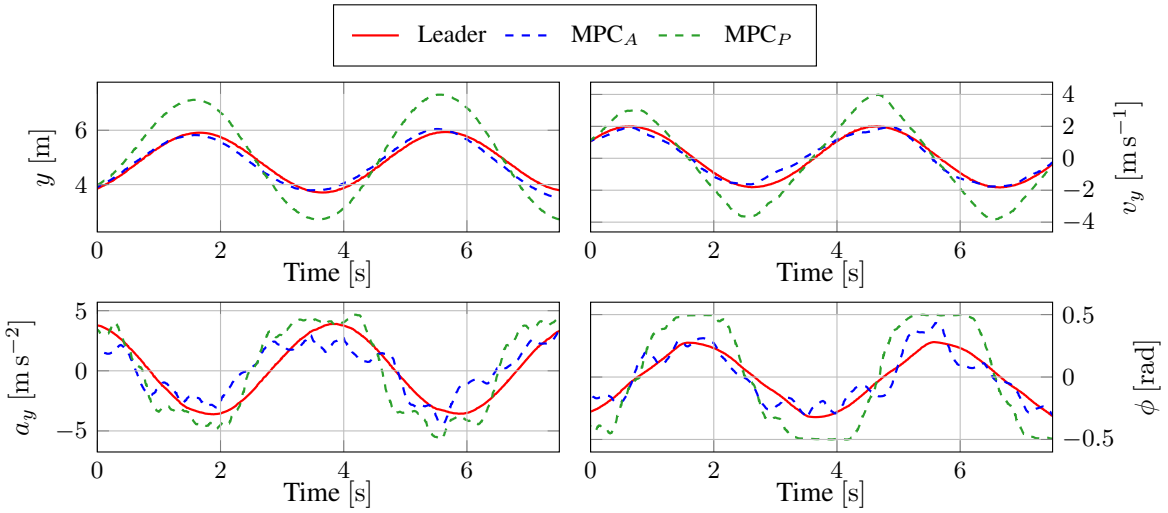


Figure 7.8: The comparison of both controllers for $dt = 2\,\text{s}$.

The statistical tracking errors of both tracking controllers are summarized in Table 7.1. As the trajectory speed increases, the performance of both controllers deteriorates. However, the MPC_P's decline in performance during more agile movements is much more significant; its tracking error worsens by approximately a factor of four when the dt is halved, compared to a factor of around two for MPC_A. Furthermore, MPC_A consistently achieves better tracking across all speeds tested, achieving approximately 80% improvement in the mean tracking error during the highest trajectory speed. The improvement is less significant at slower trajectory

speeds, where the movements of the leader UAV are less abrupt, but still present, achieving an improvement of almost 40% during $dt = 4\,\text{s}$. The exact overall improvement is shown in Table 7.2.

	MPC_A			MPC_P		
dt[s]	mean(e_y)[m]	std(e_y)[m]	$\Delta_{max}(e_y)$[m]	mean(e_y)[m]	std(e_y)[m]	$\Delta_{max}(e_y)$[m]
2	0.142	0.097	0.319	0.735	0.392	1.230
3	0.106	0.073	0.362	0.458	0.243	0.859
4	0.080	0.056	0.256	0.128	0.076	0.314

Table 7.1: Statistical results of the tracking performance of both controllers at different trajectory speeds. From left to right: the mean, standard deviation and maximum of the position error along the y-axis. The position error is calculated as the absolute difference of the leader's and follower's position $e_y = |y_L - y_F|$.

dt[s]	mean(e_y) Improvement[%]	std(e_y) Improvement[%]	$\Delta_{max}(e_y)$ Improvement[%]
2	80.68	75.26	74.07
3	76.86	69.96	57.86
4	37.50	26.32	18.47

Table 7.2: Improvement of MPC_A mean, standard deviation and maximum of the position error along the y-axis compared to MPC_P.

7.3.2 Noise Impact

To evaluate the influence of the attitude measurement noise on the performance of the controllers, we chose 4 different noise levels shown in Table 7.3. Throughout the following tests, the frequency of the attitude measurements was fixed at 50 Hz. The UAV flew the same trajectory under each noise level, with noise introduced artificially at each controller iteration. The parameters of both the LKF and MPC remained constant at the previously described values.

	σ_A[rad]	σ_A[deg]	σ_A^2[rad^2]
1	0.0	0.0	0.0
2	0.1	5.73	0.01
3	0.173	9.91	0.03
4	0.316	18.16	0.1

Table 7.3: Tested attitude noise levels. From left to right: the standard deviation in radians, the standard deviation in degrees, and the covariance in radians squared.

We show the result just for the trajectory speed $dt = 2\,\text{s}$. In Figure 7.9, the comparison of the Kalman Filter estimates for the different noise levels is shown. Across all tested covariances, the estimates of the LKF_A track the ground truth values much more accurately than LKF_P, effectively enabling a more reliable prediction for the controller. The comparison of the controller performance is depicted in Figure 7.10. A statistical summary of the tracking error for the different noise levels for all trajectory speeds is presented in Table 7.4.

Notably, even under the highest noise tested, the performance of MPC$_A$ remains significantly better than that of MPC$_P$ during agile trajectories. This result underscores the benefit of including the attitude measurements, as even with heavy noise, the use of the noisy attitude measurements outperforms the position-only based approach. During the slowest tested trajectory speed, MPC$_A$ with the noise level given by $\sigma_A^2 = 0.1\,\text{rad}^2$ achieved worse performance than MPC$_P$. In this case, the benefit of attitude knowledge was insufficient to compensate for the impact of the large noise level, degrading the controller's performance.

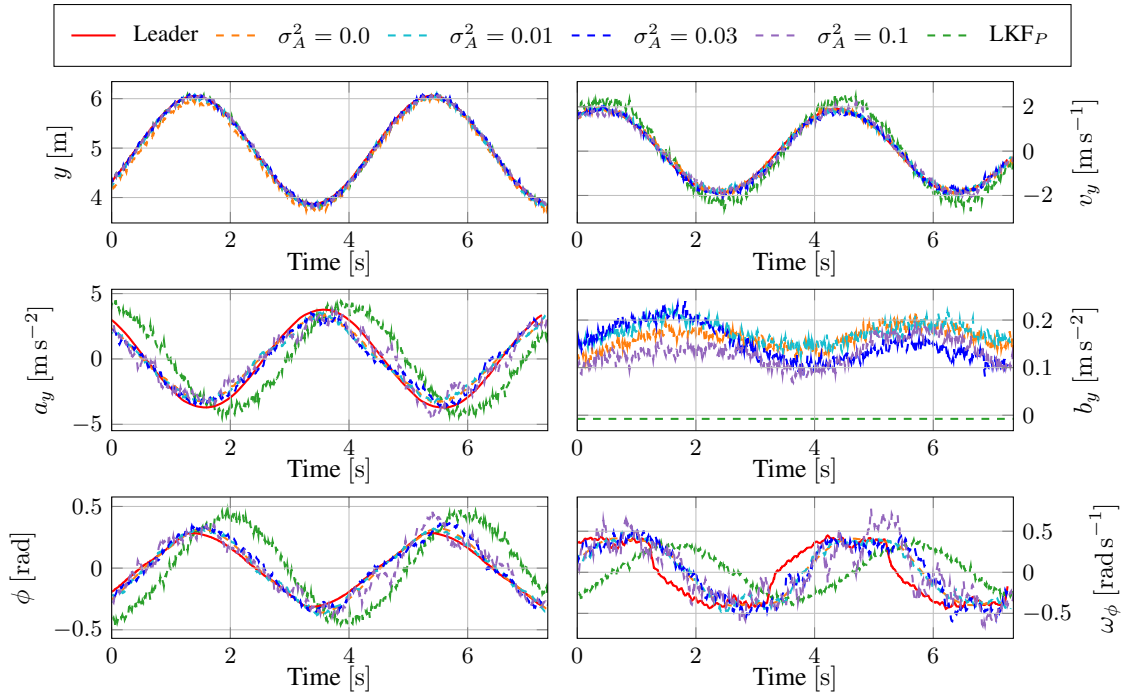


Figure 7.9: Comparison of LKF$_A$ estimates across different noise covariances for trajectory speed $dt = 2\,\text{s}$. The covariance are in radians squared. The estimates of LKF$_P$ are provided for reference.

	mean(e_y)[m]				
dt[s]	$\sigma_A^2 = 0.0$	$\sigma_A^2 = 0.01$	$\sigma_A^2 = 0.03$	$\sigma_A^2 = 0.1$	MPC$_P$
2	0.129	0.119	0.142	0.239	0.735
3	0.080	0.068	0.106	0.197	0.458
4	0.075	0.063	0.080	0.132	0.128

Table 7.4: Mean absolute tracking error in the y-axis across noise levels and speeds. The data for LKF$_P$ are also provided for reference purposes. The covariances are in radians squared.

7.3.3 Refresh Rate Impact

The influence of different measurement refresh rates was evaluated for four different values: 50 Hz, 20 Hz, 10 Hz, and 1 Hz. During the experiments, we fixed the attitude noise covariance at $\sigma_A^2 = 0.03\,\text{rad}^2$. The parameters in LKF and MPC remained fixed throughout all the experiments.

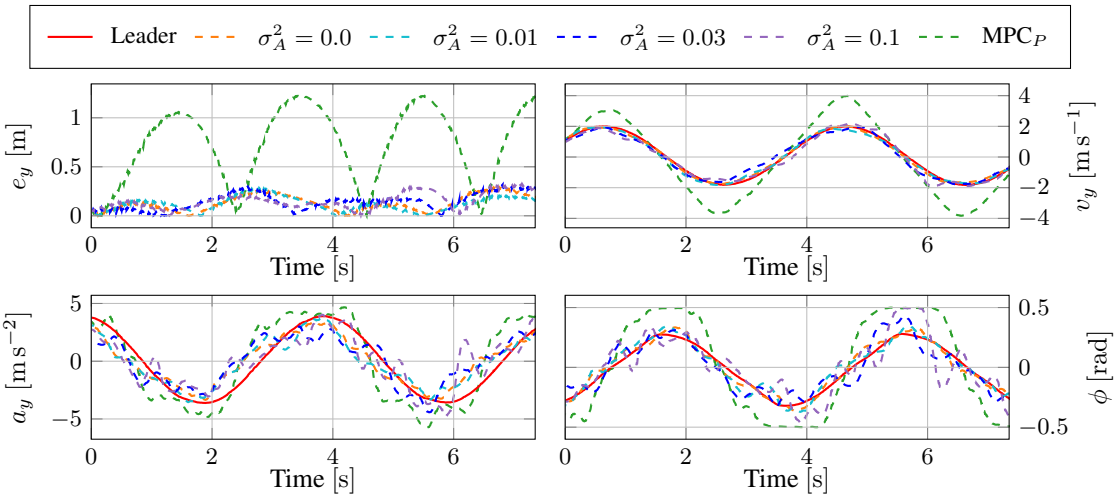


Figure 7.10: Performance of MPC$_A$ controller under varying attitude noise levels, compared with MPC$_P$, for trajectory speed $dt = 2\,\text{s}$. The covariances are in radians squared.

We show the impact on the estimates of the LKF$_A$ for $dt = 2\,\text{s}$ in Figure 7.11. Notably, the LKF$_A$ maintains an accurate estimation performance down to a refresh rate of 10 Hz. However, at the rate of 1 Hz, the estimates begin to diverge significantly, often performing worse than the LKF$_P$. This is especially evident on the acceleration plot, where the delay of the estimate is far greater. Furthermore, the times when the measurements are received are visibly seen in the acceleration and roll plots, which display a "step-like" behavior at each measurement update.

The quality of the estimates becomes evident in the controller performance, as shown in Figure 7.12. The performance of MPC$_A$ greatly excels that of MPC$_P$, with the refresh rate as low as 10 Hz. However, as was the case in the LKF estimates, at 1 Hz, the MPC$_A$'s performance deteriorates, resulting in a significantly larger tracking error and tracking delay. Additionally, the performance of the controller becomes highly dependent on the specific timing of the updates, as can be seen in the error graph, where the error for 1 Hz is first lower than that of the MPC$_P$, but worsens over time.

The statistical data for all the refresh rates, and all tested trajectory speeds, are provided in Table 7.5. The results show that at extremely low refresh rates, the attitude measurements no longer provide an advantage (since our trajectory is periodic with a whole-second period, the impact of the low refresh rate is less severe; however, for a different trajectory, it would lead to significantly worse performance). Nevertheless, an update rate of 10 Hz already brings substantial improvements, performing comparably to the highest tested rate of 50 Hz.

	mean(e_y)[m]				
dt[s]	50 Hz	20 Hz	10 Hz	1 Hz	MPC$_P$
2	0.142	0.136	0.167	0.582	0.735
3	0.106	0.118	0.123	0.556	0.458
4	0.080	0.071	0.117	0.258	0.128

Table 7.5: Comparison of the mean absolute error in position along the y-axis for the different measurement update rates.

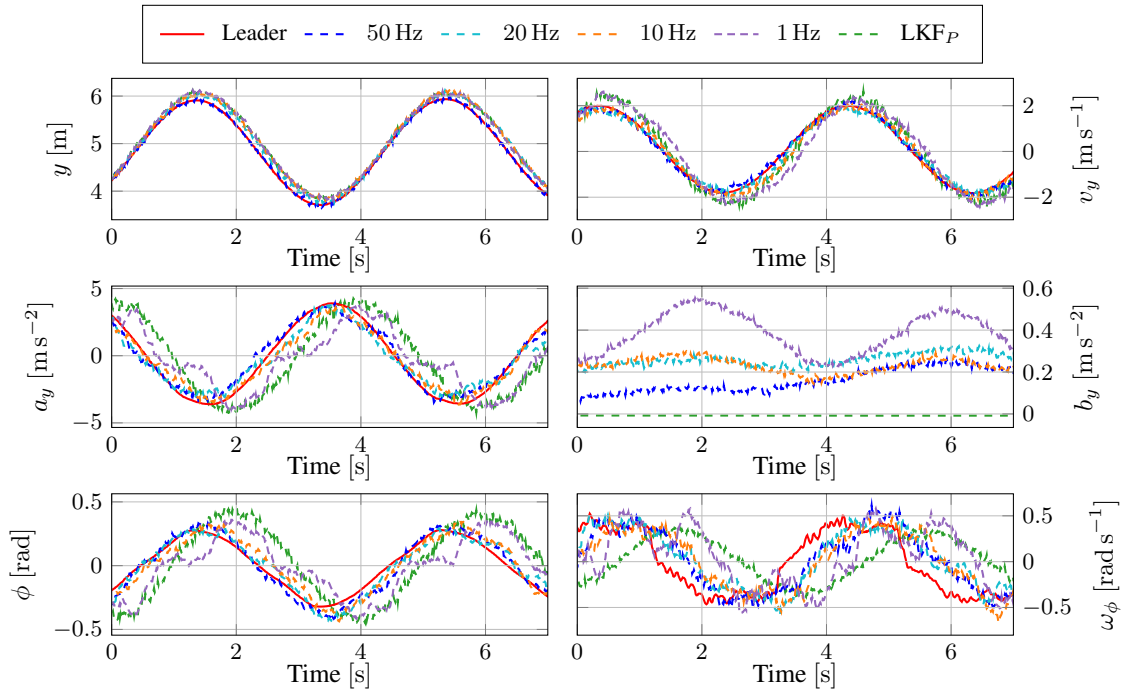


Figure 7.11: Comparison of LKF$_A$ estimates for different measurement refresh rates, with estimates of LKF$_P$ provided for reference.

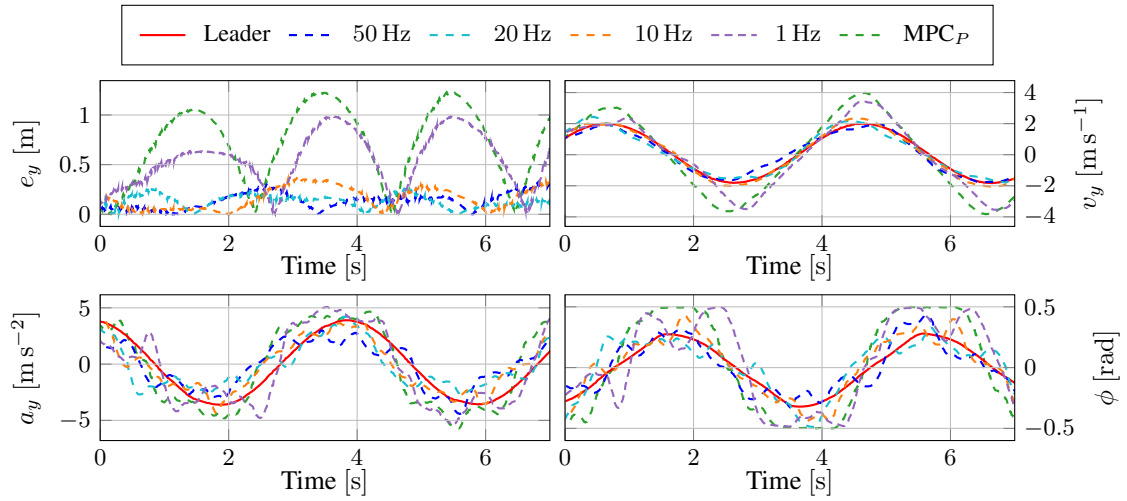


Figure 7.12: The influence of the measurement refresh rate on the performance of MPC$_A$ controller.

7.4 Real-life Experiments

To demonstrate the functionality of the control algorithm beyond simulations, tests on real hardware were conducted. The algorithm was tested on the same trajectory as used in the simulation experiments, with trajectory speed given by $dt = 3\,\text{s}$.

The controller ran on the Intel NUC onboard computer at a stable refresh rate of 100 Hz. The states of the follower UAV were obtained from the onboard estimator. For the vertical position (and thus vertical velocity and acceleration), the RTK-GPS was used, while for the horizontal, we utilized GPSGarmin[3]. The leader measurements were also obtained through the estimator, and sent as a ROS-topic to the follower UAV through Wi-Fi. Since the Wi-Fi's speed and response time varied largely, the data sometimes came with a delay or failed to arrive altogether. To handle this, we introduced a timeout mechanism on the measurement data. Every message containing the leader UAV's position and rotation measurements older than 0.5 s was discarded. If such a message arrived, the last obtained measurements were utilized in the control loop.

The parameters of the LKF and MPC were equal to the ones described in chapter 6, with the exception of three MPC cost variables, which were retuned after the real-world experiment had ended in order to improve the controller performance. To compare the real-life results with simulations, we ran the same experiments with identical parameters in the simulation environment. The MPC cost parameters used during the experiments are provided in Table 7.6.

		x	y	z	v_x	v_y	v_z	ϕ	θ	ψ	T	ϕ_R	θ_R	T_R	
Q	diag([	10	10	500	5	5	5	0	0	0	0	2000	2000	10	])
		$\Delta\phi_R$		$\Delta\phi_R$		$\Delta\phi_R$									
R	diag([	100000		100000		10000		])							

Table 7.6: The MPC cost matrices used during the real-life experiments. Red highlights the values that differ from the parameters defined in Table 6.4.

7.4.1 Results

The results, alongside their simulation counterparts, are depicted in Figure 7.13. The behavior of the real-world drones is displayed in the left column, while the simulation data are shown in the right column. Around time $t \approx 25\,\text{s}$, the leader controller during the real-life experiment briefly experienced faulty behavior, resulting in a different trajectory. The MPC$_A$'s superior tracking ability is visibly seen, tracking tightly the position, as well as velocity and acceleration. By contrast, MPC$_P$ achieves an overshoot of $\approx 1\,\text{m}$, compared to $\approx 0.4\,\text{m}$ with MPC$_A$. The velocity and acceleration tracking of MPC$_P$ are also worse, with the roll angle achieving saturation due to the unforeseen braking maneuvers.

Comparing both the real-world and simulated results reveals a strong correlation. Both controllers behaved similarly at all times (excluding the disturbance, which we were unable to model), and achieved almost the same overshoots in both scenarios. This confirms the simulation's reliability as a representation of the system dynamics and validates the algorithm's

[3]The use of RTK-GPS for the horizontal estimation came with noticeable drift in the position.

usability in real-world conditions, successfully handling delayed measurements and sometimes even complete loss of measurement data.

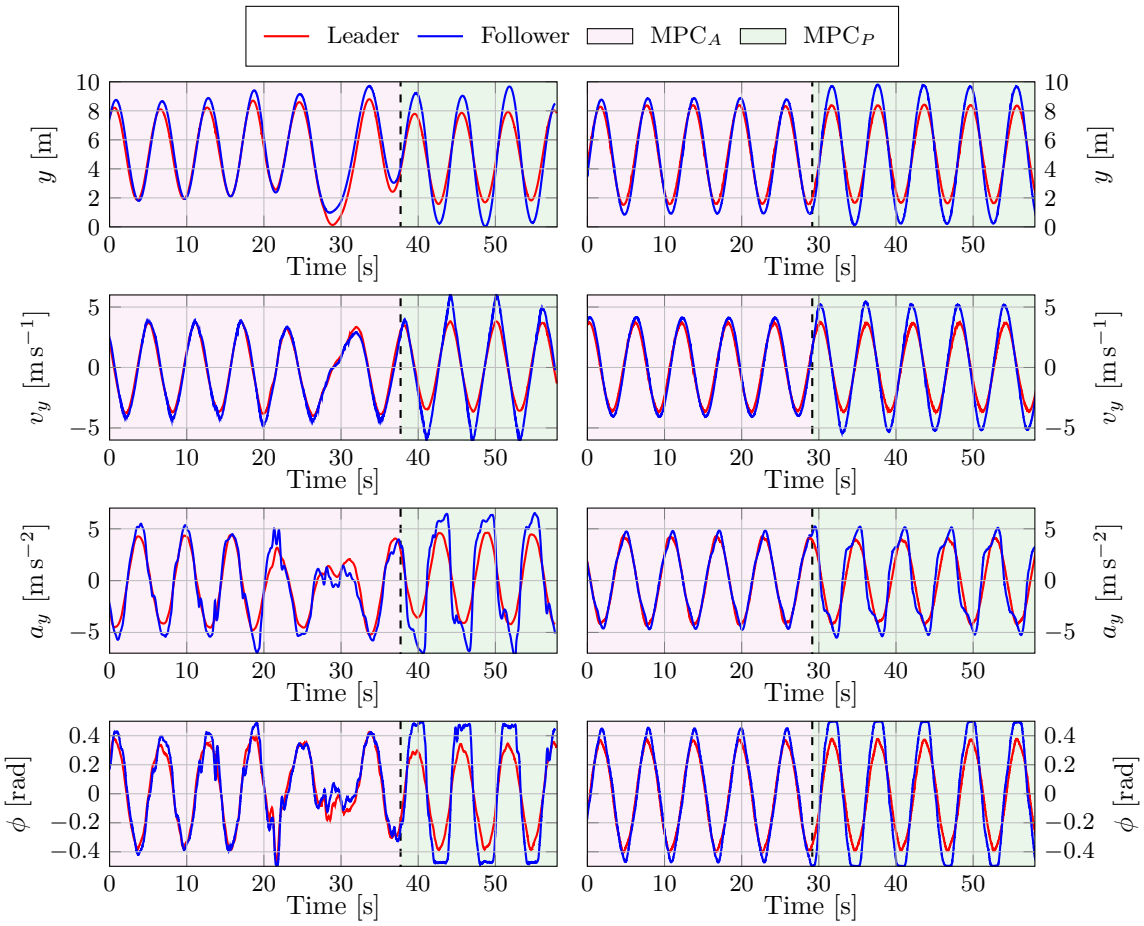
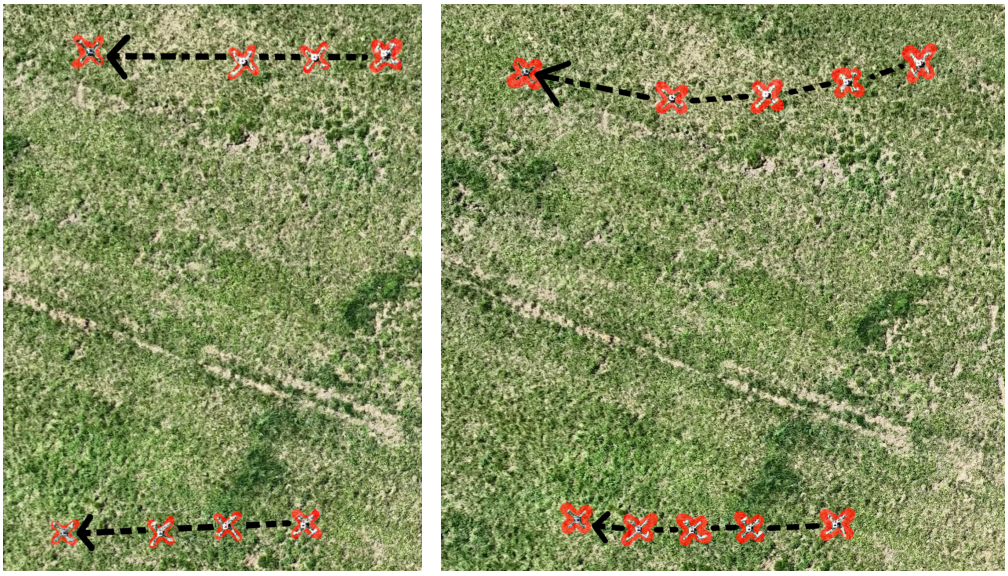


Figure 7.13: Results of the real-world experiment. In the left column, the data from the real-world experiment, in the right column, the data with the identical parameters from a simulation experiment. The use of MPC$_A$ and MPC$_P$ controllers is denoted by pink and green background, respectively.

To visualize the results more clearly, we show a motion trail composite of both UAVs utilizing both controllers in Figure 7.14. The motion shown contains one instance of the line trajectory, the same trajectory contained in the graphs shown above. The difference between the controllers is apparent, as MPC$_A$ achieves a much smaller overshoot in the braking part at the end of the line.

7.5 Summary

This section provided a comprehensive evaluation of the impact of attitude measurements on the estimation of the leader states, prediction of the short-term trajectory, and the overall performance of the control algorithm. The improved performance of the controller incorporating attitude measurements was demonstrated, particularly dominant during more agile leader movements given by faster trajectory speeds. Furthermore, the effect of the attitude measurement noise and refresh rate was analyzed. The results confirmed that attitude



(a) MPC_A. (b) MPC_P.

Figure 7.14: Motion trail composite of the line trajectory for MPC_A controller in (a), and MPC_B in (b). In both the images, the leader is located at the bottom, while the follower is at the top. The leader's left-most shown position always corresponds to the end of the braking motion, where the UAV has zero velocity. The arrow indicates the direction of flight.

measurements significantly improve performance, with a mean position error improvement of up to 80% during the highest tested trajectory speed. Even with high measurement noise and refresh rates as low as 10 Hz, the attitude measurements provided noticeable improvement.

Finally, we presented data from real-world experiments, illustrating the algorithm's effectiveness outside simulations. The data were comparable with the simulation experiments, which proved that the simulation is a valid reference for the performance of the control algorithm.

8 Conclusion

This thesis presented the design and implementation of a control algorithm for a one-to-one leader-follower UAV formation that incorporates the leader's attitude measurements to improve the tracking performance. Firstly, the UAV dynamics were studied, from which a linear mathematical model was derived as a foundation for the algorithm's components. A LKF was developed to estimate the leader's full state from noisy position and attitude measurements. The filter was extended to predict the future short-term trajectory of the leader UAV. To track the predicted trajectory, a QP-MPC controller was implemented. The whole pipeline was successfully implemented in C++ as a MRS module.

Comprehensive experiments, conducted both in simulation and in the real world, demonstrated that incorporating the attitude measurements significantly enhances the estimation of the leader's states. This led to improved short-term trajectory prediction and the overall tracking capabilities of the controller. This improvement was especially evident during agile trajectories, which involved sudden directional changes. The effect of different noise levels and the update rate of the measurements was also evaluated, highlighting the system's robustness, as it achieved a great improvement in tracking even with noisy data and low update rates.

8.1 Future Work

Future development will focus on extending the algorithm to handle more aggressive and faster flight movements. This would require retuning the MPC (and perhaps the LKF), as the current parameters leave room for further improvements. Possibly, highly agile motion may necessitate moving beyond the linear approximations, potentially requiring a nonlinear version of the Kalman Filter (such as UKF) and NMPC, as the deviation from the equilibrium point would be far greater. However, this would require more computing power for the online optimization. Incorporating real measurements of both the position and attitude, without requiring communication between the leader and the follower, also constitutes an interesting extension of this work.

9 References

[1] Z. Kewei, L. Zongzhe, Z. Xiaolin, and Z. Boxin, "Dynamic Multi-UAV Cooperative Reconnaissance Task Assignment Based on ICNP," in *2020 5th International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, Dec. 2020, pp. 773–779. DOI: 10.1109/ICMCCE51767.2020.00170. [Online]. Available: https://ieeexplore.ieee.org/document/9421463 (visited on 01/26/2025).

[2] J. Liu, Y. Liu, M. Cong, Z. Wang, and J. Wang, "A Novel Method for Multi-UAV Cooperative Reconnaissance Mission Planning in Denied Environment," in *2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC)*, Nov. 2021, pp. 1–5. DOI: 10.1109/ISCSIC54682.2021.00012. [Online]. Available: https://ieeexplore.ieee.org/document/9644320 (visited on 01/26/2025).

[3] M. Saska, V. Vonásek, J. Chudoba, J. Thomas, G. Loianno, and V. Kumar, "Swarm Distribution and Deployment for Cooperative Surveillance by Micro-Aerial Vehicles," en, *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 469–492, Dec. 2016, ISSN: 1573-0409. DOI: 10.1007/s10846-016-0338-z. [Online]. Available: https://doi.org/10.1007/s10846-016-0338-z (visited on 01/26/2025).

[4] T. Tomic, K. Schmid, P. Lutz, *et al.*, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 46–56, Sep. 2012, Conference Name: IEEE Robotics & Automation Magazine, ISSN: 1558-223X. DOI: 10.1109/MRA.2012.2206473. [Online]. Available: https://ieeexplore.ieee.org/document/6290694 (visited on 01/26/2025).

[5] M. Abdelkader, S. Güler, H. Jaleel, and J. S. Shamma, "Aerial Swarms: Recent Applications and Challenges," en, *Current Robotics Reports*, vol. 2, no. 3, pp. 309–320, Sep. 2021, ISSN: 2662-4087. DOI: 10.1007/s43154-021-00063-4. [Online]. Available: https://doi.org/10.1007/s43154-021-00063-4 (visited on 01/26/2025).

[6] *Controller Design and Disturbance Rejection of Multi-Quadcopters for Cable Suspended Payload Transportation Using Virtual Structure | IEEE Journals & Magazine | IEEE Xplore*. [Online]. Available: https://ieeexplore.ieee.org/document/9950250 (visited on 01/26/2025).

[7] I. Sa and P. Corke, "Vertical Infrastructure Inspection Using a Quadcopter and Shared Autonomy Control," en, in *Field and Service Robotics: Results of the 8th International Conference*, K. Yoshida and S. Tadokoro, Eds., Berlin, Heidelberg: Springer, 2014, pp. 219–232, ISBN: 978-3-642-40686-7. DOI: 10.1007/978-3-642-40686-7_15. [Online]. Available: https://doi.org/10.1007/978-3-642-40686-7_15 (visited on 01/26/2025).

[8] Z. Gu, B. Song, Y. Fan, and X. Chen, "Design and Verification of UAV Formation Controller based on Leader-Follower Method," in *2022 7th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Jul. 2022, pp. 38–44. DOI: 10.1109/CACRE54574.2022.9834161. [Online]. Available: https://ieeexplore.ieee.org/document/9834161 (visited on 01/26/2025).

[9] J. Devey, E. Shahra, W. Hao, D. Mi, A. Aneiba, and M. Idrissi, "Design and Simulation of a Novel Leader-Follower UAV Cluster and Formation Control Network," in *2024 International Joint Conference on Neural Networks (IJCNN)*, ISSN: 2161-4407, Jun. 2024, pp. 1–6. DOI: 10.1109/IJCNN60899.2024.10650159. [Online]. Available: https://ieeexplore.ieee.org/document/10650159 (visited on 01/26/2025).

[10] P. Shukla, S. Shukla, and A. K. Singh, "Trajectory-Prediction Techniques for Unmanned Aerial Vehicles (UAVs): A Comprehensive Survey," *IEEE Communications Surveys & Tutorials*, pp. 1–1, 2024, Conference Name: IEEE Communications Surveys & Tutorials, ISSN: 1553-877X. DOI: 10.1109/COMST.2024.3471671. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10701056?casa_token=Mshj86eA5moAAAAA:5ZzHgyltXPH61KhUlYfHeJMPnR-Ce5IqXDI58YWrtNBCI2cN3aSp5MipSP9d4JJhVqkJAnb8oVA (visited on 01/23/2025).

[11] Z. Niu, X. Jia, and W. Yao, "Communication-Free MPC-Based Neighbors Trajectory Prediction for Distributed Multi-UAV Motion Planning," *IEEE Access*, vol. 10, pp. 13 481–13 489, 2022, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3148145. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9698180 (visited on 01/23/2025).

[12] V. Tyurin, O. Martyniuk, V. Mirnenko, P. Open'ko, and I. Korenivska, "General Approach to Counter Unmanned Aerial Vehicles," in *2019 IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD)*, Oct. 2019, pp. 75–78. DOI: 10.1109/APUAVD47061.2019.8943859. [Online]. Available: https://ieeexplore.ieee.org/document/8943859 (visited on 01/26/2025).

[13] M. Pliska, M. Vrba, T. Báča, and M. Saska, "Towards safe mid-air drone interception: Strategies for tracking capture," *Robotics and Automation Letters*, vol. 9, no. 10, pp. 8810–8817, 2024. DOI: 10.1109/LRA.2024.3451768.

[14] M. Z. A. Rashid, F Yakub, H. N. M. Shah, and M. S. M. Aras, "Comprehensive review on controller for leader-follower robotic system," en, *INDIAN J. MAR. SCI.*, vol. 48, no. 07, 2019.

[15] F. Fahimi, "Sliding-Mode Formation Control for Underactuated Surface Vessels," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 617–622, Jun. 2007, ISSN: 1941-0468. DOI: 10.1109/TRO.2007.898961. [Online]. Available: https://ieeexplore.ieee.org/document/4252162 (visited on 04/25/2025).

[16] D. A. Mercado, R. Castro, and R. Lozano, "Quadrotors flight formation control using a leader-follower approach," in *2013 European Control Conference (ECC)*, Jul. 2013, pp. 3858–3863. DOI: 10.23919/ECC.2013.6669637. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6669637 (visited on 04/25/2025).

[17] W. Jasim and D. Gu, "Robust Team Formation Control for Quadrotors," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 4, pp. 1516–1523, Jul. 2018, ISSN: 1558-0865. DOI: 10.1109/TCST.2017.2705072. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7981341 (visited on 04/25/2025).

[18] T. Xu, J. Liu, Z. Zhang, G. Chen, D. Cui, and H. Li, "Distributed MPC for Trajectory Tracking and Formation Control of Multi-UAVs With Leader-Follower Structure," *IEEE Access*, vol. 11, pp. 128 762–128 773, 2023, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3329232. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10308607 (visited on 04/25/2025).

[19] K. A. Ghamry and Y. Zhang, "Formation control of multiple quadrotors based on leader-follower method," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2015, pp. 1037–1042. DOI: 10.1109/ICUAS.2015.7152394. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7152394 (visited on 04/25/2025).

[20] S. Li, Y. Wang, J. Tan, and Y. Zheng, "Adaptive RBFNNs/integral sliding mode control for a quadrotor aircraft," *Neurocomputing*, vol. 216, pp. 126–134, Dec. 2016, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2016.07.033. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216307780 (visited on 04/25/2025).

[21] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a Quadrotor With Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, Oct. 2017, ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2720851. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7961277 (visited on 04/25/2025).

[22] E. Kayacan and R. Maslim, "Type-2 fuzzy logic trajectory tracking control of quadrotor vtol aircraft with elliptic membership functions," *IEEE/ASME Transactions on Mechatronics*, vol. PP, pp. 1–1, Sep. 2016. DOI: 10.1109/TMECH.2016.2614672.

[23] J. Ghommam, H. Mehrjerdi, and M. Saad, "Leader-follower formation control of nonholonomic robots with fuzzy logic based approach for obstacle avoidance," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, ISSN: 2153-0866, Sep. 2011, pp. 2340–2345. DOI: 10.1109/IROS.2011.6094413. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6094413 (visited on 04/25/2025).

[24] C. G. Prevost, A. Desbiens, and E. Gagnon, "Extended Kalman Filter for State Estimation and Trajectory Prediction of a Moving Object Detected by an Unmanned Aerial Vehicle," in *2007 American Control Conference*, ISSN: 2378-5861, Jul. 2007, pp. 1805–1810. DOI: 10.1109/ACC.2007.4282823. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4282823?casa_token=VOQCU-KqknsAAAAA:Cb6brgRVtgQyi-hKpfe9XpYQ70Ah-f24vKDhxk7Dr8MPyJsSmDFhR6X3O3uKIkYm4OGMgZnzz9g (visited on 01/23/2025).

[25] Y. Zhang, Z. Jia, C. Dong, Y. Liu, L. Zhang, and Q. Wu, "Recurrent LSTM-based UAV Trajectory Prediction with ADS-B Information," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, ISSN: 2576-6813, Dec. 2022, pp. 1–6. DOI: 10.1109/GLOBECOM48099.2022.10000919. [Online]. Available: https://ieeexplore.ieee.org/document/10000919 (visited on 01/26/2025).

[26] P. Shu, C. Chen, B. Chen, *et al.*, "Trajectory prediction of UAV Based on LSTM," in *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, Sep. 2021, pp. 448–451. DOI: 10.1109/ICBASE53849.2021.00089. [Online]. Available: https://ieeexplore.ieee.org/document/9696074 (visited on 01/23/2025).

[27] Z. Chen, D. Guo, and Y. Lin, "A Deep Gaussian Process-Based Flight Trajectory Prediction Approach and Its Application on Conflict Detection," en, *Algorithms*, vol. 13, no. 11, p. 293, Nov. 2020, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 1999-4893. DOI: 10.3390/a13110293. [Online]. Available: https://www.mdpi.com/1999-4893/13/11/293 (visited on 01/23/2025).

[28] Z. Ma, H. Gong, and X. Wang, "Trajectory Tracking Prediction of Multiple-UAVs Formation Based on Gated Cyclic Convolution Neural Network," in *2023 8th International Conference on Automation, Control and Robotics Engineering (CACRE)*, Jul. 2023, pp. 84–90. DOI: 10.1109/CACRE58689.2023.10208829. [Online]. Available: https://ieeexplore.ieee.org/document/10208829 (visited on 01/23/2025).

[29] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," en,

[30] R. Girgis, F. Golemo, F. Codevilla, *et al.*, *Latent Variable Sequential Set Transformers For Joint Multi-Agent Motion Prediction*, arXiv:2104.00563 [cs], Feb. 2022. DOI: 10.48550/arXiv.2104.00563. [Online]. Available: http://arxiv.org/abs/2104.00563 (visited on 01/26/2025).

[31] T. Baca, G. Loianno, and M. Saska, "Embedded Model Predictive Control of Unmanned Micro Aerial Vehicles," in *IEEE International Conference on Methods and Models in Automation and Robotics (MMAR)*, IEEE, 2016, pp. 992–997.

[32] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Transactions on Automatic Control*, vol. 45, no. 3, pp. 477–482, Mar. 2000, ISSN: 1558-2523. DOI: 10.1109/9.847726. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/847726 (visited on 05/10/2025).

[33] D. Axehill, "Controlling the level of sparsity in MPC," *Systems & Control Letters*, vol. 76, pp. 1–7, Feb. 2015, ISSN: 0167-6911. DOI: 10.1016/j.sysconle.2014.12.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167691114002680 (visited on 05/10/2025).

[34] T. Baca, M. Petrlik, M. Vrba, *et al.*, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent & Robotic Systems*, vol. 102, no. 26, pp. 1–28, 1 May 2021.

[35] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Sep. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/1389727 (visited on 05/10/2025).

[36] R. Verschueren, G. Frison, D. Kouzoupis, *et al.*, *Acados: A modular open-source framework for fast embedded optimal control*, arXiv:1910.13753 [math], Nov. 2020. DOI: 10.48550/arXiv.1910.13753. [Online]. Available: http://arxiv.org/abs/1910.13753 (visited on 05/10/2025).

A Appendix A - Digital Content

In Table A.1, the names and descriptions of the included digital files are listed.

Name	Description
thesis.pdf	Bachelor's thesis in pdf format
leader_follower_controller.zip	C++ source codes for the controller
experiment.mp4	down-scaled video of the real-life experiment
experiment_link.txt	YouTube link for the experiment video in full resolution

Table A.1: Digital Content.