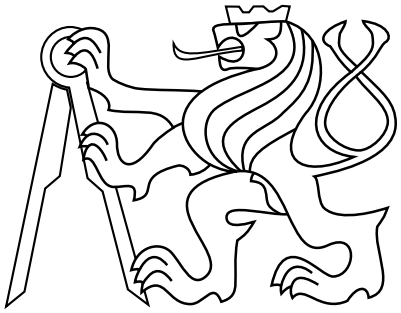
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Trajectory Planning for Autonomous Landing of a Multirotor Helicopter on a Boat

Master's Thesis

Ondřej Procházka

Prague, January 2023

Study programme: Cybernetics and Robotics

Supervisor: Ing. Tomáš Báča, Ph.D.

Author statement:

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 10.1.2023 Ondřej Procházka

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Procházka Ondřej** Personal ID number: **474744**

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

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

Trajectory Planning for Autonomous Landing of a Multirotor Helicopter on a Boat

Master’s thesis title in Czech:

Plánování trajektorie pro autonomní přistání vícerotorové helikoptéry na loď

Guidelines:

The thesis aims to tackle the problem of path and trajectory planning for autonomous landing of a multirotor Unmanned Aerial Vehicle (UAV) on top of a catamaran Unmanned Surface Vehicle (USV). UAV helicopters might be soon become useful in autonomous scanning and gathering litter from the surface of water bodies. Autonomous UAV docking onboard a small ship becomes a challenging and vital sub-task of such a large endeavor. This thesis will focus on the problem of designing an approach and safe landing strategy that could cope with the nonlinear motion of a boat on a wavy water surface. The thesis consists of the following tasks:

• Familiarize yourself with the MRS UAV System [1] for control, estimation, and simulation of multirotor helicopters.
• Study the existing approaches of autonomous trajectory generation and planning for autonomous landing of a UAV on top of a moving USV.
• Prepare a realistic simulation environment in the Gazebo/ROS simulator that includes a catamaran boat. Prepare a path-following feedback controller for the boat in order to automate testing during development.
• Define a set of estimated states (e.g., the USV’s position, velocity) that are needed for autonomous landing on the boat. The requirements will be conveyed to a fellow student, the designer of a state predictor for the USV.
• Design and implement a method for automated path and trajectory generation for autonomous landing of a quadrotor UAV on top of a moving catamaran USV.
• Verify the proposed system using the simulation environment. If possible, given the current social situation, and if the hardware allows, conduct experiments using a real-world robotic setup.

Bibliography / sources:

[1] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert and Martin Saska. 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 102(26):1–28, May 2021.
[2] Tomas Baca, Petr Stepan, Vojtech Spurny, Daniel Hert, Robert Penicka, Martin Saska, Justin Thomas, Giuseppe Loianno and Vijay Kumar. Autonomous Landing on a Moving Vehicle with an Unmanned Aerial Vehicle. Journal of Field Robotics 36(5):874-891, 2019.

Name and workplace of master’s thesis supervisor:

Ing. Tomáš Báča, Ph.D. Multi-robot Systems FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **25.01.2022** Deadline for master's thesis submission: **10.01.2023**

Assignment valid until: **30.09.2023**

Ing. Tomáš Báča, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master'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 master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Ing. Tomáš Báča, Ph.D., for his invaluable guidance, support, and encouragement throughout the creation of my diploma thesis. I am also grateful for my fellow students and colleagues from Multi-robot Systems group's invaluable support, especially with real-world experiments. Finally, I would like to extend my sincere thanks to my family and friends for their love and support. Their unwavering belief in me has been a constant source of motivation and strength, and I am forever grateful for their presence in my life.

Abstract

This thesis focuses on the design, implementation and verification of trajectory planning for the landing of an unmanned multirotor helicopter on the deck of a moving boat. The work builds on an already existing system that provides accurate estimation and prediction of vessel's states, as well as a system of precise autonomous control of an Unmanned Aerial Vehicle (UAV). The main focus of this work lies in designing a trajectory generator that uses Unmanned Surface Vehicle (USV)'s states and creates trajectories for the unmanned helicopter. The concept of Model Predictive Control, which takes advantage of an optimisation problem solver based on first-order solution methods, is utilised in the process. This allows the use of a relatively accurate model of the UAV to describe its dynamics. Thanks to that, it is possible to include position, orientation, linear velocities and Euler rates in the planning of the trajectory. The trajectory generator is capable of creating trajectories from take-off to touch-down while all the computations are performed onboard the UAV in real-time. The overall system's functionality is verified in different simulations and real-world experiments.

Keywords Unmanned Aerial Vehicles, Unmanned Surface Vehicles, Trajectory Planning, Model Predictive Control, Landing

Abstrakt

Tato práce se zabývá návrhem, implementací a ověřením systému plánování trajektorií pro přistání bezpilotní multirotorové helikoptéry na palubě plující lodi. Práce navazuje na již existující systém, který poskytuje odhad a predikci stavů lodi, a také na systém přesného autonomního řízení bezpilotní helikoptéry. Hlavní náplní této práce je návrh generátoru trajektorií, který implementuje sledování stavů lodi a vytváří trajektorie pro bezpilotní prostředek. V procesu je využit koncept prediktivního řízení (Model Predictive Control), který k řešení optimalizačních problémů využívá metody prvního řádu. To umožňuje použití relativně složitého modelu bezpilotního prostředku pro popis jeho dynamiky. Diky tomu je možné do plánování trajektorie zahrnout pozici, úhel natočení, lineární rychlost a rychlost změny Eulerových úhlů. Generátor trajektorií je schopný vytvářet trajektorie od vzletu až po přistání, zatímco všechny výpočty jsou prováděny na palubě bezpilotního prostředku a probíhají v reálném čase. Funkcionalita celkového systému je ověřována v simulacích i během reálných experimentů.

Klíčová slova Bezpilotní Prostředky, Bezpilotní Plavidla, Plánování Trajektorií, Automatické Řízení, Přistávání

Abbreviations

ACADO Automatic Control and Dynamic Optimization

ADMM Alternating Direction Method of Multipliers

AHC Active Heave Compensation

BLDC Brushless Direct Current

COG Center Of Gravity

CPU Central Processing Unit

CTU Czech Technical University

DOF Degree of Freedom

ECI Earth-centered inertial

ECEF Earth-centered Earth-fixed

EKF Extended Kalman Filter

ENU East North Up

ESC Electronic Speed Controller

FFT Fast Fourier Transformation

FOV Field of View

GLF Generalised Logistic Function

GNSS Global Navigation Satellite System

GPS Global Positioning System

GPU Graphics Processing Unit

HW Hardware

IBVS Image-Based Visual Servoing

IEKF Iterated Extended Kalman Filter

IMU Inertial Measurement Unit

LED Light-Emitting Diode

LKF Linear Kalman Filter

LMPC Linear Model Predictive Control

LPI Landing Period Indicator

LQR Linear Quadratic Regulator

LTI Linear time-invariant

LiDAR Light Detection and Ranging

MOI Moment Of Inertia

MPC Model Predictive Control

MRS Multi-robot Systems

NED North East Down

NMPC Nonlinear Model Predictive Control

ODE Ordinary Differential Equation

OpenCV Open Source Computer Vision Library

OSQP Operator Splitting Quadratic Program

PBVS Position-Based visual servoing

PID Proportional-Integral-Derivative

PWM Pulse Width Modulation

QP Quadratic Programming

RMSE Root Mean Square Error

ROS Robot Operating System

RPM Revolutions Per Minute

SAR Search and Rescue

SPA Signal Prediction Algorithm

UAV Unmanned Aerial Vehicle

UKF Unscented Kalman Filter

USV Unmanned Surface Vehicle

UV UltraViolet

UVDAR UltraViolet Direction And Ranging

VRX Virtual RobotX

VSL Visual Slide Landing

WAM-V Wave Adaptive Modular Vessel

Contents

List of Figures

List of Tables

Chapter 1

Introduction

There has been a significant development in the domain of Unmanned Aerial Vehicles (UAVs) in the past decades. Electronic components have been reduced in size, and the capacity of batteries has increased as well as their energy density. Thanks to technological advancements, powerful on-board computers provide good performance for demanding control algorithms that ensure precise and agile manoeuvrability. Therefore, drones have enormous potential in the public and industrial realm. The general public is the most familiar with UAVs that are made for capturing photos and videos. These multirotor helicopters are equipped with a high-resolution camera and can fly long distances. Even if these UAVs have functionalities like "follow me" or "predefined shot", a pilot is still required to issue these commands. However, as was mentioned before, powerful hardware, which allows for an implementation of planning algorithms for generating flight path based on surrounding world observation is available. This enables the operation of fully autonomous UAVs, which can carry out sophisticated tasks, such as surveillance of certain areas or searching for missing persons [1].

Search and Rescues (SARs) multirotor helicopters are specialised UAVs equipped with professional sensors and use self-determining evaluation systems for autonomous operation [2]. Moreover, these drones can be used for monitoring people's offshore safety and warn against sharks or identify swimmers in distress [3]. An illustration of such a drone can be seen in Fig. 1.1. These drones can also be used to detect garbage on the water surface that can be picked up by the drone itself or information about the waste's location can be shared with an Unmanned Surface Vehicle (USV) which secures the trash collection [4]. The UAVs are designed to operate mainly above the sea without the possibility of landing on land, hence the presence of a mother ship is required in the vicinity. This means that the UAVs must be able to take-off and land on the boat's deck, even in case of large waves. Since the take-off rarely represents many complications, this thesis focuses on trajectory planning for the autonomous landing of a multirotor helicopter on a boat.

Figure 1.1: Shark spotter. Adopted from [5].

1.1 State of the art

In order to study the landing on a floating vessel, it is necessary to start with definition of the word "landing", which is an act or process carried out by the multirotor UAV that leads to contact with the ground or with a determined platform for touch-down. In a standard situation, when the drone is operating above land and landing can be performed almost anywhere, it is not a problem to execute such a manoeuvre even without knowing the exact altitude and height above the ground. The UAV uses a set predefined descent speed and the touch-down is determined by monitored output thrust. This allows it to detect when the drone entirely sits on the ground only from the force which is required for hover [6].

If the UAV utilizes a sensor, which provides information about the drone's height, the landing can be performed more precisely with regard to descending speed. Unfortunately, both manoeuvres mentioned above can not ensure that the landing will be done in a suitable place. As a solution, the Global Navigation Satellite System (GNSS) is typically used outdoors to provide the drone's exact location. Based on the knowledge of the drone's precise position, a fail-safe "return to home" can be implemented to ensure that the drone will land in the same place where the take-off was carried out. Of course, the spot for the landing can be chosen wherever since that the Earth's surface is well documented and the safe spots for landing can be selected. However, this method still suffers from low accuracy and weak or interrupted signal from the GNSS.

To increase precision of the UAV's landing, a camera or a stereo camera can be mounted to the UAV to obtain visual information about the landing spot. Thanks to these sensors, data for computer vision is available and therefore it is possible to attach a recognisable visual landmark on the landing site, which can improve the stability and accuracy for autonomous landing even if the Global Positioning System (GPS) is not available. On heliports, a big letter H or X is commonly used, however, this does not provide as much information as the usage of special tags. In [7], the proposal of a coloured marker with Vision-based position and attitude estimation for a UAV is presented. Similarly, the AprilTag is a black and white flexible visual reference system that has many methods of use and therefore it is more studied in [8]. Nevertheless, both tags can be used as a reference to obtain UAV's orientation and location relatively to the landing pad and thus this information can be used in control algorithms. In [9], the algorithm for real-time estimation of the landing pad using AprilTag is presented. Moreover, Proportional-Integral-Derivative (PID) controller described in [10] was used for vision-based autonomous landing which calculates with a USV without roll, pitch and heave motions. Therefore, the USV must float straight in constant speed and as smooth as possible.

Other method, which uses Open Source Computer Vision Library (OpenCV) to detect ArUrco markers [11] for robust and accurate landing on a moving target, is presented in [12]. Using two tags enables utilisation of the first marker to guide the UAV during the approach and tracking whereas the second tag is placed on the helipad. The Visual Slide Landing (VSL) approach enables landing on a boat, in a situation where it's not possible to hover above the helipad and perform a regular landing because of sails or cables.

Information from the vision sensors can be used to control the UAV, thus the Position-Based visual servoing (PBVS) and Image-Based Visual Servoing (IBVS) are mentioned in [13]. The PBVS method works in Cartesian space and whereas the IBVS approach works with the image input from sensors directly. The IBVS suffers from camera calibration errors, however, the PBVS may suffer from position estimation errors. Moreover, it is worth mentioning that the method assumes that the UAV's attitude changes are small and therefore can be neglected.

Furthermore, this method uses future estimation of the landing spot for smoother control, using the Kalman filter. The output from the IBVS controller is yaw rate, descent speed and linear lateral velocities. The solution described above was tested using a real world deck simulation platform which proves that the controller is capable to land in such conditions.

As long as the UAV's computer is capable of computing the estimated state and position of the USV in real-time, it can be assumed that the Central Processing Unit (CPU) is powerful enough to also calculate a prediction of the USV's states in the future. The information about the position or even the knowledge of movements of the boat in the future can be used by a method called Model Predictive Control (MPC), which is also known as receding horizon control or moving horizon control [14], [15]. The use of the MPC finds application in many projects. One of the most closely related projects could be landing on a moving car [16]. This study may be used as a starting point for this thesis, but naturally, many other aspects, like heave and side slips caused by waves, must be studied further. Landing on a floating vessel is more closely studied in [17]. Moreover, there already exists a research that implements not only Linear Model Predictive Control (LMPC) but also Nonlinear Model Predictive Control (NMPC) for helicopters [18].

Expansion of the aforementioned approaches for the UAV's precise landing on the boat's deck can be found in [19]–[21]. This work summarises usage of Signal Prediction Algorithm (SPA), Active Heave Compensation (AHC) and Landing Period Indicator (LPI). The SPA includes mode detection, estimation and prediction of the USV's states, which uses Fast Fourier Transformation (FFT) to identify frequency, amplitude and phase of the roll and pitch movements. During the landing phase the algorithm increases and decreases the prediction horizon based on the estimated time to landing and also validates the size of the roll, pitch movements using thresholds. On the other hand, LPI uses the estimation of the ship's energy to determine an opportunity to land. To increase safety during landing the AHC is proposed in the same papers. The approach of this algorithm is to stabilise the UAV above the deck to ensure a smooth landing taking into account raising and lowering of the deck, therefore it can be used together with SPA or LPI. Moreover, the same algorithm is used to estimate time to landing. Unfortunately, these methods are not perfect and have their limitations, which are derived from the parameters of the boat used for the experiment (30 m long, with 5.9 m beam, a draft of 2 m and mass of 218 t) and thresholds for roll ($\pm 5°$) and pitch ($\pm 2°$). To determine the ship's movements, three Light Detection and Rangings (LiDARs) sensors are used, which is highly dependent on the boat being in the Field of View (FOV), and there is no verification that the followed object is in fact the intended vessel. On the other hand, the UAV was able to follow and land on a Navy patrol vessel moving roughly $5\,\mathrm{m\,s^{-1}}$ (10 kn) in $10\,\mathrm{m\,s^{-1}}$ wind.

Due to the fact, that the weather and wind conditions at sea are often severe, the UAV must be capable of flight in such conditions. It is not only more challenging to fly in such conditions, but bigger and more frequent waves are formed. The dynamic landing on moving platform in turbulent wind conditions is studied in [22].

Last but not least it is necessary to mention that also the boat itself can participate in the landing. For example, Wave Adaptive Modular Vessel (WAM-V) boats are equipped with joints on its floats which reduces rocking caused with waves. Moreover, in [17] the UAV and the USV collaborate to reach their goal.

1.2 Contributions

This thesis presents a customisable trajectory generator for a precise landing of a multirotor helicopter on top of a moving vessel. The system is fully autonomous and capable of creating a trajectory for the UAV from take-off to touch-down in real time. The trajectory generator is built on MPC based full-state tracker. Therefore, the generated trajectories already include the UAV's dynamics. As an internal model of the MPC, a linear model of the UAV consisting of twelve states is being used. Thanks to that, not just position but also orientation, linear velocities and Euler rates are taken into account during the creation of the full-state requirement for the trajectory generator. Nevertheless, the system would not work without the prediction of the USV's states. These states are being used during the final approach manoeuvre to reduce the harshness of impact of the UAV's legs on the deck. Moreover, the attitude angles of the UAV can be aligned with the USV's. The presented trajectory generator was incorporated into the Multi-robot Systems (MRS)'s UAV control system and tested in realistic simulations. The system was also deployed in real-world experiments.

1.3 Outline

This thesis consists of seven major chapters, excluding Introduction chapter (Chap. 1) and Conclusion chapter (Chap. 9). The Chap. 2 which follows after the chapter Introduction is called Preliminaries. This chapter describes all preliminaries, from the mathematical notation, over the definition of the coordinate frame and modelling of the USV, to the MRS UAV control system, on which this thesis relies. The next chapter, Chap. 3, introduces the concept of trajectory tracker based on MPC. The chapter also contains a mathematical derivation of the objective function. Chap. 4 describes the UAV's nonlinear mathematical model. This chapter also includes the identification of the parameters that describe the UAV. The trajectory creation is introduced in Chap. 5. The Chap. 6 is devoted to the development tools which were used to develop and verify the proposed solution in this thesis. These development tools include ROS, MATLAB, Gazebo simulator and quadratic programming solvers. The simulation validations are described in Chap. 7. The last Chap. 8, is devoted to real-world experiments and includes images taken during those experiments.

Chapter 2

Preliminaries

This chapter is dedicated to introducing the foundations, that are used as a basis for the trajectory planner designed in this thesis. Therefore, common reference frames (Sec. 2.2), mathematical modelling of the USV (Sec. 2.3.1), description of the USV's states estimation and prediction (Sec. 2.3.2), description of the MRS UAV control pipeline (Sec. 2.4) and hardware used in real-world experiments (Sec. 2.5) are presented in the following sections.

2.1 Definitions

2.1.1 Mathematical notation

The mathematical notation, nomenclature and notable symbols used in this thesis are shown in the table Table 2.1.

$\mathbf{x}$, $\boldsymbol{\alpha}$	vector, pseudo-vector, or tuple
$\hat{\mathbf{x}}$, $\hat{\boldsymbol{\omega}}$	unit vector or unit pseudo-vector
$\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$	elements of the *standard basis*
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{I}$	identity matrix
$\mathbf{0}$	zero matrix
$x = \mathbf{a}^\intercal \mathbf{b}$	inner product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x} = \mathbf{a} \times \mathbf{b}$	cross product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x}_{(n)} = \mathbf{x}^\intercal \hat{\mathbf{e}}_n$	n^{th} vector element (row), $\mathbf{x}, \mathbf{e} \in \mathbb{R}^3$
$\mathbf{X}_{(a,b)}$	matrix element, (row, column)
x_d	x_d is *desired*, a reference
$\dot{x}, \ddot{x}, \dddot{x}$	1^{st}, 2^{nd}, and 3^{rd} time derivative of x
$x_{[n]}$	x at the sample n
SO(3)	3D special orthogonal group of rotations
SE(3)	*SO(3)* $\times \mathbb{R}^3$, special Euclidean group
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integers
$\mathbf{x}^\top, \mathbf{x}^\intercal$	transposition of the vector $\boldsymbol{x}$
$diag([a, b, \ldots, z])$	diagonal matrix composed of $a, b, \ldots, z$ on its diagonal
$\mathbf{X} = \mathbf{A} \hat{\circ} \mathbf{B}$	takes matrix $\mathbf{A}$ and multiplies every element of the matrix $\mathbf{B}$
$\mathcal{W}$	world reference frame
$\mathcal{C}, \mathcal{B}$	body-fixed multirotor and body-fixed vessel frames of reference
$\mathbf{R}_w^b$	rotation matrix from the world frame to the vessel's body-fixed frame
$\mathbf{R}_w^c$	rotation matrix from the world frame to the UAV's body-fixed frame
$\mathbf{Q}$	objective function matrix (associated with process states)
$\mathbf{P}$	terminal penalty weight matrix (associated with process states)
$\mathbf{R}$	objective function matrix (associated with process inputs)
ϕ, θ, ψ	roll, pitch and yaw angles
h	heading
t	time
k	iteration step

Table 2.1: Mathematical notation, nomenclature and notable symbols.

2.1.2 Path

Path is a set of waypoints defined as $\{(x_1, y_1, z_1, h_1), \ \ldots, (x_N, y_N, z_N, h_N)\}$, where $N \in \mathbb{Z}^+$ of points which define the position and heading in 3D space.

2.1.3 Trajectory

Trajectory is a path regularly sampled at Δt time increments between individual points.

2.1.4 Linearity

A function $f(x)$ is linear if and only if the Additivity and Homogeneity are satisfied:

$$f(x + y) = f(x) + f(y), \tag{2.1}$$
$$f(\alpha x) = \alpha f(x). \tag{2.2}$$

2.1.5 Linear time-invariant system

Linear time-invariant (LTI) system is a system which meets the condition of linearity and which is time-invariant. The system can be introduced in a discrete form as

$$\mathbf{x}_{[t+1]} = \mathbf{A}\mathbf{x}_{[t]} + \mathbf{B}\mathbf{u}_{[t]}, \tag{2.3}$$
$$\mathbf{y}_{[t]} = \mathbf{C}\mathbf{x}_{[t]} + \mathbf{D}\mathbf{u}_{[t]}, \tag{2.4}$$

where $\mathbf{x}_{[t]} \in \mathbb{R}^n$ is the state vector at the sample t, $\mathbf{u}_{[t]} \in \mathbb{R}^m$ is the input vector, $\mathbf{y}_{[t]} \in \mathbb{R}^n$ is the output vector, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the main system's dynamics matrix, $\mathbf{B} \in \mathbb{R}^{n \times m}$ is the system input matrix, $\mathbf{C} \in \mathbb{R}^{n \times n}$ represents output matrix and $\mathbf{D} \in \mathbb{R}^{n \times m}$ is a feed-forward matrix.

2.1.6 Solvers for Model Predictive Control

Solvers for MPC are online numerical algorithms for solving MPC problems.

2.2 Common reference frames

Since we operate in the open three-dimensional Euclidean space, defining various coordinate frames of reference is crucial for increasing transparency and comprehensibility. Within these systems, the mathematical model of the UAV and USV will be defined.

According to [23], it is convenient to define Earth-centered inertial (ECI) and Earth-centered Earth-fixed (ECEF) coordinate frames. ECI is a non-accelerating reference frame placed at the centre of Earth in which Newton's laws of motion apply. On the contrary, ECEF's origin remains fixed to the Earth's centre, but axes rotate relatively to the inertial frame ECI. Therefore, the latter is usually used for global navigation and guidance.

The following coordinate system is defined relative to the Earth's reference ellipsoid, and its origin's position is determined with longitude and latitude angles in ECEF. The geographic reference frame is defined as North East Down (NED) or East North Up (ENU). The axis of the NED points to the true north, east and directly to the Earth's centre. On the other

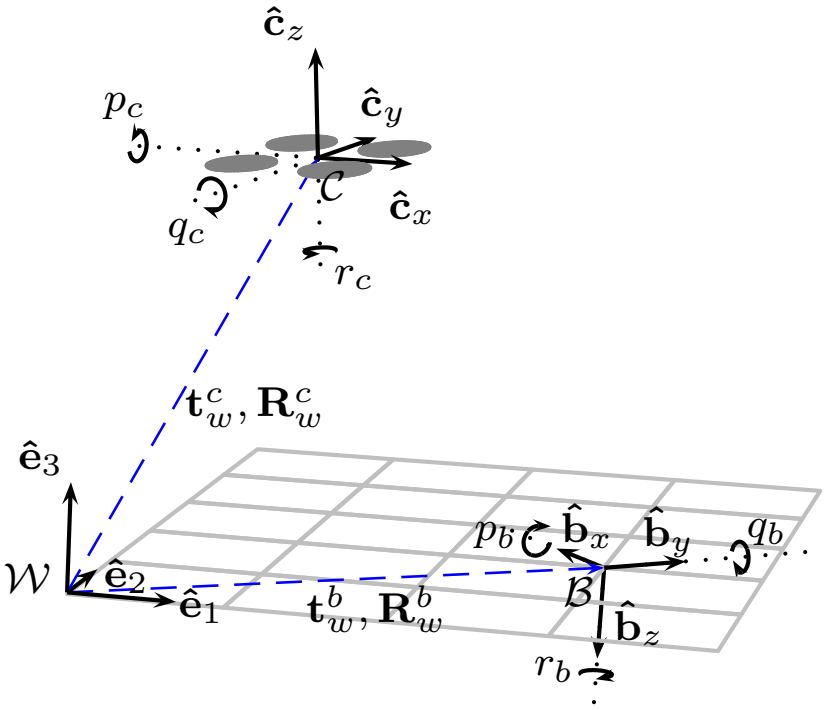


Figure 2.1: Coordinate systems illustration, where $\mathcal{W}$ represents local ENU frame, $\mathcal{C}$ denotes multirotor's body-fixed frame and $\mathcal{B}$ symbolises vessel's body-fixed frame.

hand, ENU's axes point to the east, true north and in the opposite direction from the Earth's centre. The NED frame is commonly used in the aerospace industry as well as in the marine industry. However, the ENU is predominantly used in robotics. Both coordinate frames are used for flat Earth navigation, and therefore, it is expected that the origin of this frame stays fixed in one location on the Earth's surface. Therefore, it is also necessary to assume that the operating space is limited only to the local area.

The last coordinate system which will be used is a body-fixed frame. The origin of this frame usually coincides with the object's centre of gravity, and the x-axis points to the forward direction. The directions of the remaining axes can be chosen, such as the coordinate frame will be right-handed.

The position and orientation are described relative to the local navigation frame. The linear and angular velocities should be expressed in a body-fixed frame. The illustration of used reference frames is shown in Fig. 2.1.

2.3 State estimation of an unmanned surface vessel

In the thesis we rely on the prior work of Filip Novák, who wrote the thesis "State Estimation of an Unmanned Surface Vehicle by an Unmanned Multirotor Helicopter" [24]. Therefore it is possible to build on proposed methods and ways of solving both USV estimation and USV prediction.

2.3.1 Mathematical modelling of a marine vessel

The aforementioned thesis models the boat in 6 Degrees of Freedom (DOFs), where nomenclature for the individual components can be seen in Fig. 2.2. Whereas the designation of the individual angles (roll ϕ, pitch θ, yaw ψ) and speed around individual axes (p rotation around x-axis, q rotation around the y-axis, r rotation around the z-axis) is used in the same way as for the UAV, the motion in the different axis uses different terminology. *Surge* stands for longitudinal motion, *sway* for lateral motion and *heave* for vertical motion in the body-fixed reference frame [25].

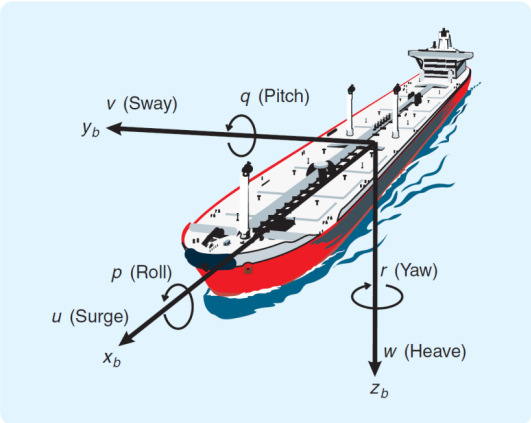


Figure 2.2: Standard notation and sign conventions for ship motion description (SNAME, 1950). Adopted from [26].

Derivation of the mathematical model was inspired by a handbook, which studies the hydrodynamic modelling and marine craft motion control systems [27]. Moreover, this literature studies the derivation of the marine craft equations of motion in great detail. Firstly, the theory around transformations is introduced, which is used to derive the kinematics part of the equation of motion. Since the dynamics' second part is kinetics, the literature deals with the analysis of the forces causing the motion. This theory is hereafter enhanced with hydrodynamic modelling.

The general form of vectorial representation of the 6 DOFs marine craft Equation of Motion is

$$\dot{\boldsymbol{\eta}} = \mathbf{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{2.5}$$

where $\boldsymbol{\eta} = [x, y, z, \phi, \theta, \psi]^{\top} \in \mathbb{R}^3 \times \mathcal{S}^3$ is a vector composed of positions and orientations, where Euler angles ϕ, θ and ψ are defined on the interval $\mathcal{S} = [0, 2\pi]$ and $\boldsymbol{\nu} = [u, v, w, p, q, r]^{\top} \in \mathbb{R}^6$ is a vector, which contains linear and angular speed for all axes. $\mathbf{J}_{\Theta}(\boldsymbol{\eta})$ is a transformation matrix from a body-fixed frame to a local frame, where subscript Θ denotes Euler angles and therefore $\mathbf{J}_{\Theta}(\boldsymbol{\eta}) \cong (\mathbf{t}_b^w, \mathbf{R}_b^w)$. The second part (kinetics) of the marine craft equation of motion has the following form:

$$\underbrace{\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu}}_{\text{rigid-body forces}} + \underbrace{\mathbf{M}_A\dot{\boldsymbol{\nu}}_r + \mathbf{C}_A(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r + \mathbf{D}(\boldsymbol{\nu}_r)\boldsymbol{\nu}_r}_{\text{hydrodynamic forces}} + \underbrace{\mathbf{g}(\boldsymbol{\eta}) + \mathbf{g}_o}_{\text{hydrostatic force}} = \boldsymbol{\tau} + \boldsymbol{\tau}_{\text{wind}} + \boldsymbol{\tau}_{\text{wave}}, \tag{2.6}$$

where

- $\mathbf{M}_{RB}$ — rigid-body system inertia matrix
- $\mathbf{C}_{RB}$ — rigid-body Coriolis and centripetal matrix
- $\mathbf{M}_A$ — hydrodynamic system inertia matrix
- $\mathbf{C}_A$ — hydrodynamic Coriolis and centripetal matrix
- $\mathbf{D}(\boldsymbol{\nu}_r)$ — damping matrix
- $\mathbf{g}(\boldsymbol{\eta})$ — vector of gravitational/buoyancy forces and moments
- $\mathbf{g}_o$ — vector used for pretrimming (ballast control)
- $\boldsymbol{\tau}$ — vector of control inputs
- $\boldsymbol{\tau}_{\text{wind}}$ — vector of wind forces
- $\boldsymbol{\tau}_{\text{wave}}$ — vector of wave-induced forces
- $\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c$ is relative velocity vector with $\boldsymbol{\nu}_c$ as the generalised ocean current velocity of an irrotational fluid

Since the Equations of motion 2.5 and 2.6 are composed in total by 12 Ordinary Differential Equations (ODEs), the order of the system is 12. This leads to a very complex system, and thus some assumptions and simplifications were made. According to [24], the simplified nonlinear system is described as

$$\dot{\boldsymbol{\eta}} = \mathbf{J}_\Theta(\boldsymbol{\eta})\boldsymbol{\nu}, \tag{2.7}$$
$$\dot{\boldsymbol{\nu}} = \mathbf{M}^{-1}(-\mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{D}(\boldsymbol{\nu}) - \mathbf{G}\boldsymbol{\eta}) + \boldsymbol{\nu}_{\text{wave}}, \tag{2.8}$$
$$\dot{\mathbf{x}}_{\text{wave},u} = \mathbf{f}_{\text{wave}}(\mathbf{x}_{\text{wave},u}), \tag{2.9}$$
$$\dot{\mathbf{x}}_{\text{wave},v} = \mathbf{f}_{\text{wave}}(\mathbf{x}_{\text{wave},v}), \tag{2.10}$$
$$\dot{\mathbf{x}}_{\text{wave},w} = \mathbf{f}_{\text{wave}}(\mathbf{x}_{\text{wave},w}), \tag{2.11}$$
$$\dot{\mathbf{x}}_{\text{wave},p} = \mathbf{f}_{\text{wave}}(\mathbf{x}_{\text{wave},p}), \tag{2.12}$$
$$\dot{\mathbf{x}}_{\text{wave},q} = \mathbf{f}_{\text{wave}}(\mathbf{x}_{\text{wave},q}), \tag{2.13}$$
$$\dot{\mathbf{x}}_{\text{wave},r} = \mathbf{f}_{\text{wave}}(\mathbf{x}_{\text{wave},r}), \tag{2.14}$$

where $\mathbf{g}(\boldsymbol{\eta}) \approx \mathbf{G}\boldsymbol{\eta}$ is a linear approximation of the restoring forces based on Archimedes' principle. $\mathbf{D}(\boldsymbol{\nu})$ represents damping of the system, $\mathbf{C}(\boldsymbol{\nu})$ denotes Coriolis and centripetal forces and $\mathbf{M} = \mathbf{M}_{RB} + \mathbf{M}_A$. $\boldsymbol{\nu}_{\text{wave}}$ is a comprehensive state vector, which contains output signal from the wave systems and $\mathbf{x}_{\text{wave}}$ includes individual states of the waves.

According to [24], the nonlinear model of the USV without the waves can be linearized under the assumption that the roll and pitch angles are small. Moreover, the waves must be described with the linear state space model. For the marine vessel, the resulting linear equations are as follows

$$\dot{\boldsymbol{\eta}}_L = \boldsymbol{\nu}, \tag{2.15}$$
$$\dot{\boldsymbol{\nu}} = -\mathbf{M}^{-1}(\mathbf{D}\boldsymbol{\nu} + \mathbf{G}\boldsymbol{\eta}_L) + \mathbf{C}_{\text{wave},\boldsymbol{\nu}}, \tag{2.16}$$
$$\dot{\mathbf{x}}_{\text{wave},\nu} = \mathbf{A}_{\text{wave},\boldsymbol{\nu}}\mathbf{x}_{\text{wave},\nu}, \tag{2.17}$$

where $\dot{\boldsymbol{\eta}}_L$ is the time derivative of the vessel's parallel coordinate system. According to [27], $\boldsymbol{\eta}_L$ is defined as

$$\boldsymbol{\eta}_L = \mathbf{J}_\psi^\intercal(\psi)\boldsymbol{\eta}, \tag{2.18}$$

which denotes the position and orientation in a global coordinate frame expressed in a body-fixed coordinate frame, where roll and pitch angles are assumed to be small (close to zero).

2.3.2 State estimation and prediction of a marine vessel

The precise USV's state estimation and prediction is a crucial requirement for planning the trajectory for landing the UAV on the boat's deck or even for approach. As mentioned in [24], the Kalman filter is used for this purpose. Kalman filter is a recursive algorithm well known since 1960 when R.E. Kalman published a paper describing discrete data linear filtering problem [28]. Thenceforth, many extensions of Kalman filters like Extended Kalman Filter (EKF) [29], Iterated Extended Kalman Filter (IEKF) and Unscented Kalman Filter (UKF) [30] were developed. The implementation and verification of the individual filters, except IEKF, was implemented in [24]. Even though the results in [24] show that the UKF has better results than the Linear Kalman Filter (LKF), the LKF was used. It is because the LKF was managed to tune better during further experiments that are not stated in the thesis mentioned above.

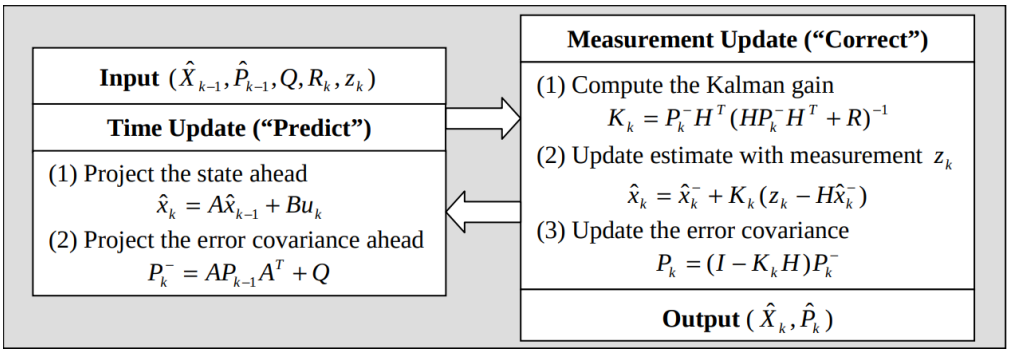


Figure 2.3: Diagram illustrating the operation of the linear Kalman filter. Adopted from [31].

The illustration of the LKF iteration steps is shown in Fig. 2.3. The measurement update step uses the information from the sensors to correct the time update step. According to [24], the following sensors were used to obtain the USV's states. GPS and an Inertial Measurement Unit (IMU) were mounted on the boat together with a computer and other necessary electronics (commonly referred to as boat unit). While the GPS information is shared using wireless communication between USV and UAV, when the boat is far away, the measurements from IMU are used during the UAV's precise landing. Since there are also AprilTag and UltraViolet Direction And Ranging (UVDAR)'s UltraViolet (UV) Light-Emitting Diodes (LEDs) on the board, the information about the boat's state is also possible to achieve by observing the USV's deck from the UAV using AprilTag or UVDAR detector [32]–[35].

The prediction step uses the vessel's mathematical model derived in Sec. 2.3.1. The estimation of the USV's state depends on the correction update step, but they are made separately. Therefore, the time update step can be used for the prediction of the USV's future states.

2.4 Multi-robot Systems multirotor control

The Multi-robot Systems (MRS) UAV control[1] was developed by the MRS group from the Czech Technical University (CTU). Since the group's field of interest is vast the control system is very general and therefore it can be used for many projects without larger modifications [36]–[39]. The description of the system architecture is derived from the definition of the MRS UAV system [6]. The control pipeline diagram is shown in Fig. 2.4.

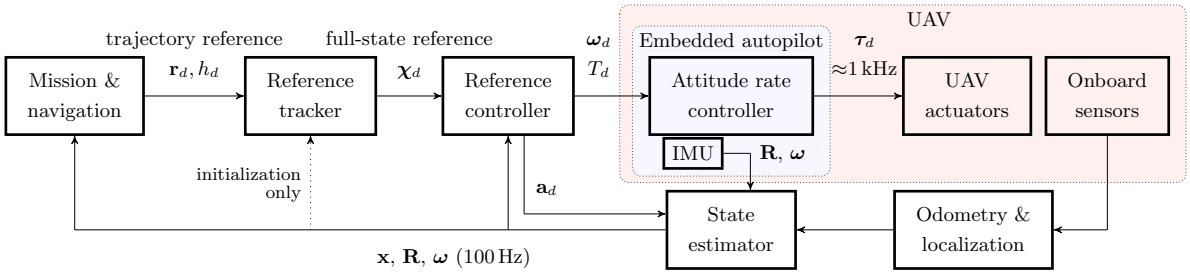


Figure 2.4: A diagram of the Multi-robot Systems multirotor control architecture. According to [6]: *Mission & navigation* software supplies the position and heading reference ($\mathbf{r}_d$, η_d) to a reference tracker. *Reference tracker* creates a smooth and feasible reference $\boldsymbol{\chi}$ for the reference feedback controller. The feedback *Reference controller* produces the desired thrust and angular velocities (T_d, $\boldsymbol{\omega}_d$) for the Pixhawk embedded flight controller. The *State estimator* fuses data from *Onboard sensors* and *Odometry & localization* methods to create an estimate of the UAV translation and rotation ($\mathbf{x}$, $\mathbf{R}$).

2.4.1 Embedded autopilot

The lowest level of the control pipeline structure is called the *Attitude rate controller*, which is responsible for controlling individual motor's Revolutions Per Minute (RPM) based on the incoming reference such that the required attitude rates of the UAV are satisfied [40]. Overall thrust T_d and angular rates $\boldsymbol{\omega}_d$ of the UAV are requested as the reference for this controller. The UAV's actual state is measured with the IMU, which measures the angular rates and acceleration in individual axes using gyroscopes and accelerometers. Everything mentioned above is part of the embedded PixHawk flight controller[2].

2.4.2 Reference controller

A *Reference controller* generates the reference for the embedded controller based on the full state description of the UAV defined as

$$\boldsymbol{\chi}_d = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \ddot{x}, \ddot{y}, \ddot{z}] \tag{2.19}$$

and which is the desired position and its derivatives up to the acceleration dynamics. The controller was created for UAVs that operate at their limits and perform agile manoeuvres. It is a nonlinear $SE(3)$ state feedback controller, which ensures precise control, fast response

[1] https://github.com/ctu-mrs
[2] https://pixhawk.org/

and fast convergence [41]. The controller uses a nonlinear model of the UAV dynamics as follows

$$\dot{\mathbf{r}} = \mathbf{v}, \tag{2.20}$$

$$m\dot{\mathbf{v}} = f\mathbf{R}\mathbf{e}_3 + mg\mathbf{e}_3, \tag{2.21}$$

$$\dot{\mathbf{R}} = \mathbf{R}\hat{\mathbf{\Omega}}, \tag{2.22}$$

$$\mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} = \mathbf{M}, \tag{2.23}$$

where $\mathbf{r} = [x, y, z]^T$ is the position, $\mathbf{R}(\phi, \theta, \psi) \in SO(3)$ is the orientation of the UAV, $\mathbf{\Omega}(p, q, r) \in \mathbb{R}^{3\times3}$ is the angular velocity in the body-fixed frame. According to [36], the hat symbol $\hat{\cdot}$ denotes the skew-symmetry operator according to $\hat{\mathbf{x}}\mathbf{y} = \mathbf{x} \times \mathbf{y}$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$. m denotes the UAV's mass, $\mathbf{J}$ is an inertia matrix, $\mathbf{M}$ denotes total moment and g denotes gravitation acceleration. f is the net thrust produced by the propellers.

2.4.3 Reference tracker

The third control layer generates the full state information about the UAV. The input reference is a trajectory, which is a set $\{(x_1, y_1, z_1, h_1), \ldots, (x_N, y_N, z_N, h_N)\}$, where $N \in \mathbb{Z}^+$ of points which define the position and heading in 3D space, regularly sampled at Δt time increments between individual points. The tracker is built on LMPC, which uses the linear model not only as the subject for the optimisation but also to accomplish a simulated control loop [42]. The graphical representation can be seen in Fig. 2.5. The LMPC was also sped up with the move blocking technique [43], which allows having a prediction horizon of length 8 seconds (40 steps) without losing the accuracy and stability and still being able to solve the optimisation problem in real-time [37]. A simulated closed loop ensures that the predicted full-state reference is an evenly-sampled reference, which contains the desired position, its derivatives up to the jerk, the heading, and the heading rate, supplied at 100 Hz.

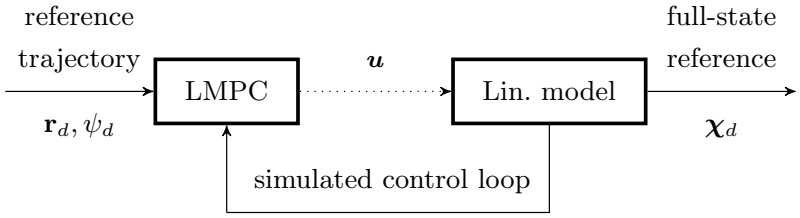


Figure 2.5: An inner interconnection of the *Reference tracker*.

2.4.4 Mission & navigation

The *Mission & navigation* block is generally defined such that can be modified to meet the requirements for the given task freely. Nevertheless, in most situations, the block ensures the trajectory creation out of a path or a single point in the 3D space. In order to create the trajectory, which will take into account the UAV's dynamics, the polynomial trajectory generation and optimisation are used [44], [45]. The generated trajectory also satisfies the requirement that the UAV completes the path in the minimum possible time. To check that the trajectory is feasible, the rigid-body model assumption and differentially flat dynamics are used [46]. Before publishing the result, the trajectory is checked for low and high thrust, high velocities, high roll, pitch, yaw rates and high yaw angular accelerations.

2.4.5 Summary

Since the system architecture mentioned above has been used in MRS group for a while now, its implementation is very general, robust and well-tuned. Therefore, the MRS UAV system has been taken as a starting point for developing the trajectory planner, which focuses on creating the trajectory that is sent as a reference to the trajectory tracker mentioned in Sec. 2.4.3.

2.5 Hardware

2.5.1 Description of the UAV

The drone which was used in real-world experiments is built on Tarot T650 Sport frame. The motors are Brushless Direct Current (BLDC) Tarot 4114 320KV with Electronic Speed Controller (ESC) Turnigy Multistar 51A. The control core of the drone is NUC8i7BEH computer to which a Pixhawk 4 (an autopilot) with GPS either with H-RTK F9P GNSS module or with standard one is connected. Everything is powered using a LiPo 6S 8000 mAh battery pack. This is standard UAV equipment in MRS group, however, some modifications were needed due to the drone's field of operation.

The drone's design for flying over the water area is presented in the Bachelor's thesis [47], which was written by Ivan Čermák. The UAV was equipped with floats on its legs, which ensures landing on the water surface in critical situations. All electronic components were sprayed with water-resistant preparation and placed in the 3D-printed cover. For the tag's detection, the drone was equipped with the RealSense camera, and for the detecting UV LEDs, the mvBlueFOX MLC200wG camera was used. The photo of the UAV is shown in Fig. 2.6.

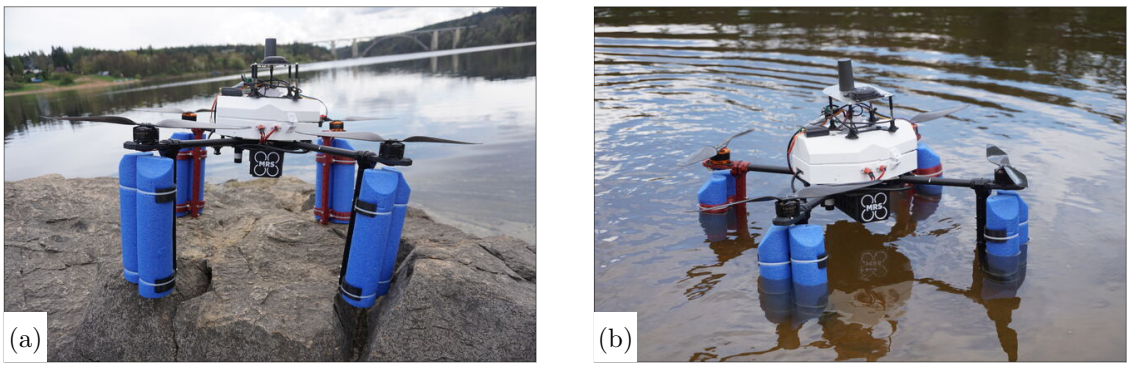


Figure 2.6: Photo of the UAV. (a) Front view. (b) UAV floating on water.

2.5.2 Description of the USV

The vessel was created out of an inflatable boat and a landing platform by the MRS group member. The landing platform was designed in the shape of a square with a side length of 2 meters. The boat unit, which contains NUC8i7BEH computer, LiPo battery pack, GPS, IMU units and other electronics, was mounted on the top of a deck, as shown in Fig. 2.7a. ApriTag and UVDAR UV LEDs were placed on the boat's deck as is shown in Fig. 2.7b. The

area of a shape of a square of the side length 80 cm was covered with AprilTag. Since the inflatable boat was not equipped with propulsion gear, the vessel with a landing platform was towed behind a powered manned vessel.

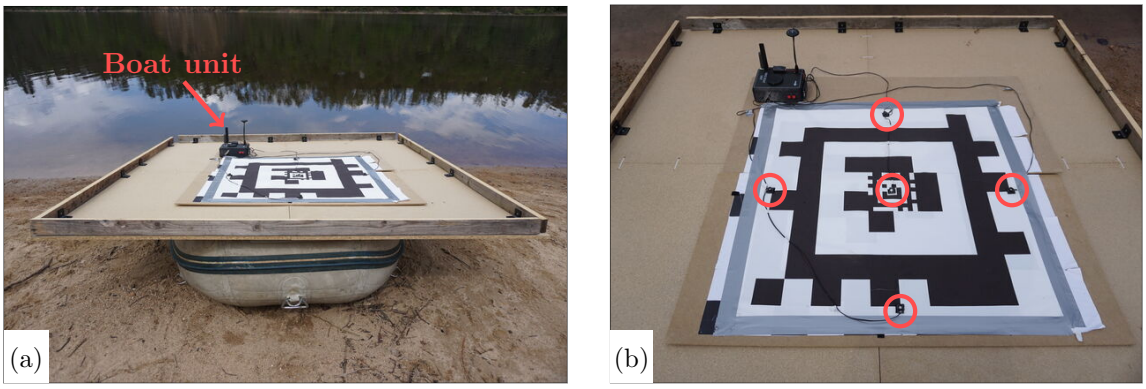


Figure 2.7: Photos of the USV. (a) Front view showing the location of the boat unit. (b) Detailed photo of the landing pattern (AprilTag) with UV LEDs marked with blue circles.

Chapter 3

Model Predictive Control

The MPC concept is known from the last century, but the most significant spread has been experienced in the past two decades [48]. This is probably due to technological advancement, which allows computations of complicated optimisation problems online. An indisputable advantage is that the MPC is universal and intuitive, which increases the field of interest enormously. A detailed overview with more than 2200 industrial applications of MPC is summarised in [49], [50]. Even classical controllers like PID or Linear Quadratic Regulator (LQR) are being replaced with MPC. Unlike common controllers, the MPC can coordinate multiple inputs and outputs and requires a mathematical model of the system's dynamics. Such a model can be linear or nonlinear.

The model constrains the system's future evolution until the end of the finite receding horizon and calculates the input signal for the future states, where the length of the computed outputs is called the control horizon. Fig. 3.1 shows the prediction horizon and control horizon in discrete form, where the number of steps is the same for both. However, the length of the prediction horizon and control horizon can be different. The prediction and control horizons length selection affect the computational complexity and the desired performance of the optimisation problem [51]. The longer prediction horizon is useful in situations where the action of the controlled system has a longer reaction time, e.g., due to the system's physical limitations like acceleration or deceleration. On the other hand, if the MPC is used for control, the input signal value does not have to be computed far into the future. The computation time is also affected by the accuracy of the result from the optimisation problem and, therefore, sometimes is preferred less accurate solution that can be computed faster.

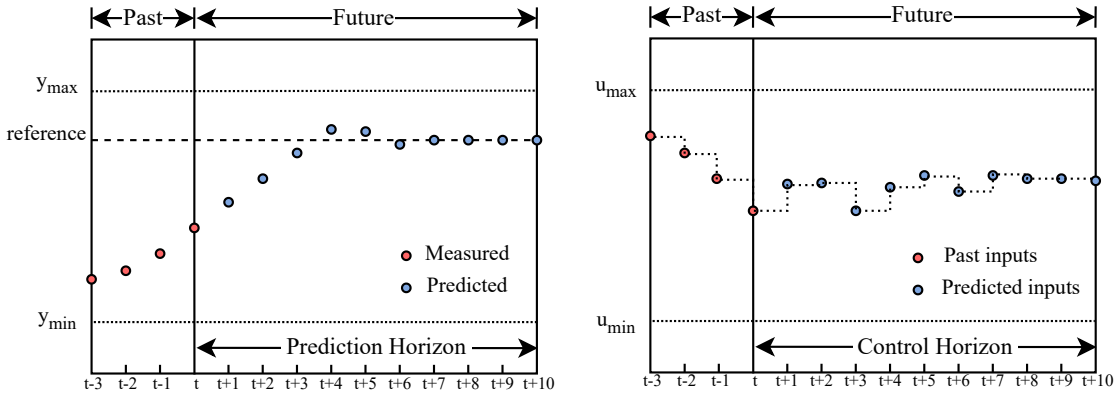


Figure 3.1: A graphical representation of the MPC algorithm.

The MPC problem is often extended with the system's physical constraints, e.g., maximal speed, attitude angle and angular rates. These inequality constraints are called hard constraints and limit the system's states or outputs within lower and upper bounds. In some

situations, it may be impossible to satisfy the hard constraints, for example, due to the disturbances, and therefore the relaxation of these bounds must be made. The relaxed hard constraints are called soft constraints [52].

Another MPC's advantage is that it can be used not only for control but also for tracking reference [53], [54]. In most cases, a reference is a point or a whole trajectory in three-dimensional space. The trajectory trackers are solely studied when the desired trajectory is in the vicinity. Therefore the solution to the optimisation problem can be found quickly. On the other hand, when the desired position is far, the solver may need much more time to find an accurate solution. Since the time duration is predominantly determined with the used solver for the optimisation problem, more information can be found in the section Sec. 6.4.

3.1 Linear Model Predictive Control

According to [55], the linear MPC problem takes the form of Quadratic Programming (QP) problem. The most general definition of the cost function for the regulation problem, where the prediction and control horizon are the same length, is

$$\min_{\mathbf{x},\,\mathbf{u}} \quad J(\mathbf{u},\mathbf{x}) = \frac{1}{2}\mathbf{x}_{[t+N]}^{\intercal}\mathbf{P}\mathbf{x}_{[t+N]} + \frac{1}{2}\sum_{k=1}^{N-1}(\mathbf{x}_{[t+k]}^{\intercal}\mathbf{Q}\mathbf{x}_{[t+k]} + \mathbf{u}_{[t+k]}^{\intercal}\mathbf{R}\mathbf{u}_{[t+k]}). \tag{3.1}$$

The model, initial state and both state and input constraints are defined as

$$\mathbf{x}_{[t+k+1]} = \mathbf{A}\mathbf{x}_{[t+k]} + \mathbf{B}\mathbf{u}_{[t+k]} \quad \forall k \in \{0,\dots,N-1\}, \tag{3.2}$$

$$\mathbf{x}_{[t]} = \text{given}, \tag{3.3}$$

$$\mathbf{x}_{\min} \leq \mathbf{x}_{[t+k]} \leq \mathbf{x}_{\max} \quad \forall k \in \{1,\dots,N\}, \tag{3.4}$$

$$\mathbf{u}_{\min} \leq \mathbf{u}_{[t+k]} \leq \mathbf{u}_{\max} \quad \forall k \in \{1,\dots,N\} \tag{3.5}$$

where $k = \{1,\dots,N-1\}$ and $N \in \mathbb{Z}^{+}$ is the size of the prediction horizon and control horizon simultaneously. $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{R}$ are penalisation, diagonal, positive semi-definite matrices. $\mathbf{R}$ is a weighting matrix penalising input signals in every control step, and $\mathbf{Q}$ is a weighting matrix penalising states in every prediction step except the last one, which is penalised with the matrix $\mathbf{P}$. The inequality constraints are defined for states with lower bound $\mathbf{x}_{\min}$ and upper bound $\mathbf{x}_{\max}$ as well as for input signal, where the lower bound is defined as $\mathbf{u}_{\min}$ and upper bound as $\mathbf{u}_{\max}$. The lower and upper bounds are given constant vectors.

3.2 Linear Model Predictive Control based trajectory tracking

Reference tracking slightly differs from the regulation problem mentioned in Sec. 3.1. Firstly, the reformulation of the reference needs to be done. Instead of bringing the system's outputs to zero, we minimise

$$\mathbf{e}_{[t+k]} = \mathbf{r}_{[t+k]} - \mathbf{y}_{[t+k]}, \quad \forall k \in \{1,\dots,N\}, \tag{3.6}$$

which is an error between the reference $\mathbf{r}$ and the system output $\mathbf{y}$. Thus the minimising function (3.1) is rewritten into the form

$$\min_{\mathbf{e},\,\mathbf{u}} \quad \frac{1}{2}\mathbf{e}_{[t+N]}^{\intercal}\mathbf{P}\mathbf{e}_{[t+N]} + \frac{1}{2}\sum_{k=1}^{N-1}\left(\mathbf{e}_{[t+k]}^{\intercal}\mathbf{Q}\mathbf{e}_{[t+k]} + \mathbf{u}_{[t+k]}^{\intercal}\mathbf{R}\mathbf{u}_{[t+k]}\right). \tag{3.7}$$

Moreover, assuming a steady state, the objective function (3.7) is unaware that the control input is not assumed to be zero. Therefore, in the minimising process, the solver tries to find a solution and minimise the input values in the same way as the tracking error. On the other hand, this can increase the deviation between the desired and actual state, which means that the reference error grows. Imagine the UAV is in hover mode (the aircraft hovers in place for a fixed time with requirements to stay in one position), the regulation error is near zero. However, the control input $\mathbf{u}_{[t]}$, which should be a required rotation of the motors, must be nonzero. Therefore, the objective function is reformulated, and instead of using the input signal itself, its increment is used as follows

$$\Delta\mathbf{u}_{[t]} = \mathbf{u}_{[t]} - \mathbf{u}_{[t-1]}. \tag{3.8}$$

This leads to the reformulation of a linear model, which must be augmented with the new state variable $\mathbf{x}_{u_{[t-1]}} \cong \mathbf{u}_{[t-1]}$ and the control input is exchanged for $\Delta\mathbf{u}_{[t]}$. The form of the augmented model is as follows

$$\begin{bmatrix}\mathbf{x}_{[t+1]}\\ \mathbf{x}_{u_{[t+1]}}\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{0}_{m\times n} & \mathbf{I}_{m\times m}\end{bmatrix}}_{\mathbf{A}_a}\underbrace{\begin{bmatrix}\mathbf{x}_{[t]}\\ \mathbf{x}_{u_{[t]}}\end{bmatrix}}_{\mathbf{x}_{a[t]}} + \underbrace{\begin{bmatrix}\mathbf{B}\\ \mathbf{I}_{m\times m}\end{bmatrix}}_{\mathbf{B}_a}\Delta\mathbf{u}_{[t]}, \tag{3.9}$$

$$\mathbf{y}_{[t]} = \underbrace{\begin{bmatrix}\mathbf{C} & \mathbf{0}_{p\times m}\end{bmatrix}}_{\mathbf{C}_a}\begin{bmatrix}\mathbf{x}_{[t]}\\ \mathbf{x}_{u_{[t]}}\end{bmatrix}, \tag{3.10}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a state vector containing n states, $\mathbf{u} \in \mathbb{R}^m$ is control vector with m inputs and $\mathbf{y} \in \mathbb{R}^p$ is a system output vector. It is also assumed that the primary model omits direct feed-through of the input signal.

3.2.1 Cost function

The optimisation cost function with an augmented model in the simultaneous form is derived from (3.7) substituting $\mathbf{e}_{[t]}$ with $\mathbf{r}_{[t]} - \mathbf{C}_a\mathbf{x}_{a[t]}$ as

$$\begin{aligned}\min_{\mathbf{x}_a,\ \Delta\mathbf{u}}\quad & \underbrace{\frac{1}{2}\mathbf{r}_{[t+N]}^\intercal\mathbf{P}\mathbf{r}_{[t+N]}}_{\text{constant term}} - \mathbf{r}_{[t+N]}^\intercal\mathbf{P}\mathbf{C}_a\mathbf{x}_{a[t+N]} + \frac{1}{2}\mathbf{x}_{a[t+N]}^\intercal\mathbf{C}_a^\intercal\mathbf{P}\mathbf{C}_a\mathbf{x}_{a[t+N]} + \dots\\ \sum_{k=1}^{N-1}\Bigg(& \underbrace{\frac{1}{2}\mathbf{r}_{[t+k]}^\intercal\mathbf{Q}\mathbf{r}_{[t+k]}}_{\text{constant term}} - \mathbf{r}_{[t+k]}^\intercal\mathbf{Q}\mathbf{C}_a\mathbf{x}_{a[t+k]} + \frac{1}{2}\mathbf{x}_{a[t+k]}^\intercal\mathbf{C}_a^\intercal\mathbf{Q}\mathbf{C}_a\mathbf{x}_{a[t+k]} + \frac{1}{2}\Delta\mathbf{u}_{[t+k]}^\intercal\mathbf{R}\Delta\mathbf{u}_{[t+k]}\Bigg).\end{aligned} \tag{3.11}$$

The equation (3.11) includes constant terms which are offsets that can be left out without losing the objective function's generality. Hence the cost function (3.11) is rewritten into the simplified matrix form as

$$\begin{aligned}J(\Delta\mathbf{u},\ \mathbf{x}_a) = \frac{1}{2}\mathbf{x}_a\underbrace{diag(\begin{bmatrix}\mathbf{C}_a\mathbf{Q}\mathbf{C}_a & \dots & \mathbf{C}_a\mathbf{Q}\mathbf{C}_a & \mathbf{C}_a\mathbf{P}\mathbf{C}_a\end{bmatrix})}_{\bar{\mathbf{Q}}}\ \dots\\ \dots\ -\mathbf{r}^\intercal\underbrace{diag(\begin{bmatrix}\mathbf{Q}\mathbf{C}_a & \dots & \mathbf{Q}\mathbf{C}_a & \mathbf{P}\mathbf{C}_a\end{bmatrix})}_{\bar{\mathbf{T}}}\mathbf{x}_a + \frac{1}{2}\Delta\mathbf{u}^\intercal\underbrace{diag(\begin{bmatrix}\mathbf{R} & \dots & \mathbf{R}\end{bmatrix})}_{\bar{\mathbf{R}}}\Delta\mathbf{u},\end{aligned} \tag{3.12}$$

where

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_{[t]}^\intercal & \mathbf{r}_{[t+1]}^\intercal & \dots & \mathbf{r}_{[t+N]}^\intercal \end{bmatrix}^\top, \tag{3.13}$$

$$\mathbf{x}_a = \begin{bmatrix} \mathbf{x}_{a[t]}^\intercal & \mathbf{x}_{a[t+1]}^\intercal & \dots & \mathbf{x}_{a[t+N]}^\intercal \end{bmatrix}^\top, \tag{3.14}$$

$$\Delta\mathbf{u} = \begin{bmatrix} \Delta\mathbf{u}_{[t]}^\intercal & \Delta\mathbf{u}_{[t+1]}^\intercal & \dots & \Delta\mathbf{u}_{[t+N-1]}^\intercal \end{bmatrix}^\top. \tag{3.15}$$

The augmented model's representation is then rewritten into the matrix form. Therefore, the update of the state vector takes the form of

$$\mathbf{x}_a = \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_a & \mathbf{0} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{A}_a & \mathbf{0} \end{bmatrix}}_{\bar{\mathbf{A}}} \mathbf{x}_a + \underbrace{\begin{bmatrix} \mathbf{B}_a & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \vdots \\ \mathbf{0} & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & \mathbf{0} & \mathbf{B}_a \end{bmatrix}}_{\bar{\mathbf{B}}} \Delta\mathbf{u} + \underbrace{\begin{bmatrix} \mathbf{A}_a \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}}_{\bar{\mathbf{A}}_{[t]}} \mathbf{x}_{a[t]}. \tag{3.16}$$

The objective function includes states $\mathbf{x}$ and inputs $\mathbf{u}$ that are coupled through the model, thus this form is called simultaneous. The simultaneous form can be simplified into the sequential form, where the cost function depends only on $\Delta\mathbf{u}$ and $\mathbf{x}_{[t]}$. Therefore, the $\mathbf{x}_a$ must be eliminated. The states vectors are lined up into the one combined vector $\bar{\mathbf{x}}_a$ whose size corresponds with the length of the prediction horizon similar to the equation denoted in [56] or [57]. The matrix form looks as follows

$$\underbrace{\begin{bmatrix} \mathbf{x}_{a[t+1]} \\ \vdots \\ \mathbf{x}_{a[t+N]} \end{bmatrix}}_{\bar{\mathbf{x}}_a} = \underbrace{\begin{bmatrix} \mathbf{B}_a & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{A}_a\mathbf{B}_a & \mathbf{B}_a & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{A}_{a[t+N-1]}\mathbf{B}_a & \dots & \mathbf{A}_a\mathbf{B}_a & \mathbf{B}_a \end{bmatrix}}_{\bar{\mathbf{C}}_a} \underbrace{\begin{bmatrix} \Delta\mathbf{u}_{[t]} \\ \vdots \\ \Delta\mathbf{u}_{[t+N-1]} \end{bmatrix}}_{\Delta\mathbf{u}} + \underbrace{\begin{bmatrix} \mathbf{A}_a \\ \vdots \\ \mathbf{A}_a^N \end{bmatrix}}_{\bar{\mathbf{A}}_a} \bar{\mathbf{x}}_{a[t]}, \tag{3.17}$$

where $N \in \mathbb{Z}^+$ is the size of the prediction horizon and $\bar{\mathbf{C}}_a$ is called the controllability matrix because every row is composed of the controllability matrix, which belongs to the individual prediction step. After substituting $\mathbf{x}_{a[t]}$ into the (3.12) the general form of the cost function is rewritten into the matrix form as

$$J(\Delta\mathbf{u},\ \mathbf{x}_a) = \frac{1}{2}\Delta\mathbf{u}^\intercal \underbrace{(\bar{\mathbf{C}}_a^\intercal \bar{\mathbf{Q}} \bar{\mathbf{C}}_a + \bar{\mathbf{R}})}_{\mathbf{H}} \Delta\mathbf{u} + \begin{bmatrix} \mathbf{x}_{a[t]}^\intercal & \mathbf{r}^\intercal \end{bmatrix} \underbrace{\begin{bmatrix} \bar{\mathbf{A}}_a^\intercal \bar{\mathbf{Q}} \bar{\mathbf{C}}_a \\ -\bar{\mathbf{T}} \bar{\mathbf{C}}_a \end{bmatrix}}_{\mathbf{F}^\intercal} \Delta\mathbf{u}. \tag{3.18}$$

3.2.2 Constraints

The cost function does not include any information about the system's limits. Therefore, to restrain the values in which the used model can operate the hard-constraints must be defined. It is possible to bound the states, outputs or even the input signal. Since the cost function was introduced in sequential form the constraints for the states must be calculated using the (3.17) as follows

$$\underbrace{\bar{\mathbf{C}}_a}_{\mathbf{G}_{\text{xmax}}} \Delta\mathbf{u} \leq \underbrace{\mathbf{x}_{\max}}_{\mathbf{W}_{\text{xmax}}} - \underbrace{\bar{\mathbf{A}}_a}_{\mathbf{S}_{\text{xmax}}} \mathbf{x}_{a[t]}, \tag{3.19}$$

where $\mathbf{x}_{\max} \in \mathbb{R}^{(n+m)\cdot N}$. Moreover, the states can also be constrained from below using the equation

$$\underbrace{-\bar{\mathbf{C}}_a}_{\mathbf{G}_{\text{xmin}}} \Delta\mathbf{u} \leq \underbrace{-\mathbf{x}_{\min}}_{\mathbf{W}_{\text{xmin}}} + \underbrace{\bar{\mathbf{A}}_a}_{\mathbf{S}_{\text{xmin}}} \mathbf{x}_{a[t]}, \tag{3.20}$$

where $\mathbf{x}_{\min} \in \mathbb{R}^{(n+m)\cdot N}$. The system's output signal can also be limited. The inequality equations are following

$$\underbrace{\mathbf{C}_a \hat{\circ} \bar{\mathbf{C}}_a}_{\mathbf{G}_{\text{ymax}}} \Delta\mathbf{u} \leq \underbrace{\mathbf{y}_{\max}}_{\mathbf{W}_{\text{ymax}}} - \underbrace{\mathbf{C}_a \hat{\circ} \bar{\mathbf{A}}_a}_{\mathbf{S}_{\text{ymax}}} \mathbf{x}_{a[t]}, \tag{3.21}$$

$$\underbrace{-\mathbf{C}_a \hat{\circ} \bar{\mathbf{C}}_a}_{\mathbf{G}_{\text{ymin}}} \Delta\mathbf{u} \leq \underbrace{\mathbf{y}_{\min}}_{\mathbf{W}_{\text{ymin}}} - \underbrace{\mathbf{C}_a \hat{\circ} \bar{\mathbf{A}}_a}_{\mathbf{S}_{\text{ymin}}} \mathbf{x}_{a[t]}. \tag{3.22}$$

The constraints on the input signal restrict the change of the input values, therefore it is commonly called the slew rate. The slew rate constraints are defined as

$$\begin{aligned} \Delta\mathbf{u}_{[t]} &\leq \Delta\mathbf{u}_{\max} \\ -\Delta\mathbf{u}_{[t]} &\leq -\Delta\mathbf{u}_{\min} \end{aligned} \Leftrightarrow \underbrace{\begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ -1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & -1 \end{bmatrix}}_{\mathbf{G}_{\text{u}}} \Delta\mathbf{u} \leq \underbrace{\begin{bmatrix} \Delta\mathbf{u}_{\max} \\ \vdots \\ \Delta\mathbf{u}_{\max} \\ -\Delta\mathbf{u}_{\min} \\ \vdots \\ -\Delta\mathbf{u}_{\min} \end{bmatrix}}_{\mathbf{W}_{\text{u}}}, \tag{3.23}$$

where $\Delta\mathbf{u}_{\max} \in \mathbb{R}^{(m,N)}$ is a vector which includes the limit values for the input signal.

It can happen that the optimisation problem may be infeasible, thus, no solution exists within the hard-constrained model. The soft constraints can be used to violate the hard-constrained limits to find the solution. To enlarge the region of the attraction, two approaches of the soft constraints are presented in [58]. Using soft constraints ensures that the MPC can find the solution for the optimisation problem. On the other hand, it increases the complexity of the problem and the time needed for convergence. Therefore, soft constraints are mentioned as a possibility that can be added at the expense of slower computation.

3.2.3 Summary

After all aforementioned computations and definitions, it is possible to introduce the general form of MPC

$$\min_{\Delta\mathbf{u}} J(\Delta\mathbf{u}, \mathbf{x}_{[t]}) = \frac{1}{2}\Delta\mathbf{u}^{\intercal}\mathbf{H}\Delta\mathbf{u} + \begin{bmatrix} \mathbf{x}_{[t]}^{\intercal} & \mathbf{u}_{[t-1]}^{\intercal} & \mathbf{r}_{[t]}^{\intercal} \end{bmatrix} \mathbf{F}^{\intercal}\Delta\mathbf{u} \tag{3.24}$$

$$\text{subject to} \quad \mathbf{G}\Delta\mathbf{u} \leq \mathbf{W} + \mathbf{S}\begin{bmatrix} \mathbf{x}_{[t]} \\ \mathbf{u}_{[t-1]} \end{bmatrix}, \tag{3.25}$$

where

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{\text{xmax}}^{\intercal} & \mathbf{G}_{\text{xmin}}^{\intercal} & \mathbf{G}_{\text{ymax}}^{\intercal} & \mathbf{G}_{\text{ymin}}^{\intercal} & \mathbf{G}_{\text{u}}^{\intercal} \end{bmatrix}^{\top}, \tag{3.26}$$

$$\mathbf{W} = \begin{bmatrix} \mathbf{W}_{\text{xmax}}^{\intercal} & \mathbf{W}_{\text{xmin}}^{\intercal} & \mathbf{W}_{\text{ymax}}^{\intercal} & \mathbf{W}_{\text{ymin}}^{\intercal} & \mathbf{W}_{\text{u}}^{\intercal} \end{bmatrix}^{\top}, \tag{3.27}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{\text{xmax}}^{\intercal} & \mathbf{S}_{\text{xmin}}^{\intercal} & \mathbf{S}_{\text{ymax}}^{\intercal} & \mathbf{S}_{\text{ymin}}^{\intercal} & \mathbf{0}_{m+n\times N\cdot m\cdot 2} \end{bmatrix}^{\top}. \tag{3.28}$$

This formulation can be used in many solvers that are able to solve quadratic problems. Sec. 6.4 is dedicated to the topic of MPC solvers, where we study and compare them rigorously. Moreover, the MPC requires a model describing the system's dynamics, therefore the derivation and identifications of the UAV model are shown in Sec. 4.

3.3 Nonlinear Model Predictive Control

Since industrial projects require more precise control and performance, the attention is turning to the Nonlinear Model Predictive Control (NMPC) nowadays [59]. The main difference between LMPC and NMPC is that the cost function can be nonquadratic and the prediction model nonlinear. Moreover, the inequality and equality constraints can also be nonlinear. According to [59], the mathematical formulation of the NMPC problem is defined to find

$$\min_{\bar{\mathbf{u}}(\cdot)} \quad J(\mathbf{x}(t), \bar{\mathbf{u}}(\cdot); T_c, T_p), \tag{3.29}$$

with objective function

$$J(\mathbf{x}(t), \bar{\mathbf{u}}(\cdot); T_c, T_p) := \int_t^{t+T_p} F(\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau)) d\tau \tag{3.30}$$

subject to:

$$\dot{\bar{\mathbf{x}}}(\tau) = \mathbf{f}((\bar{\mathbf{x}}(\tau), \bar{\mathbf{u}}(\tau)), \quad \bar{\mathbf{x}}(\tau) = \mathbf{x}(t), \tag{3.31}$$

$$\bar{\mathbf{u}}(\tau) \in \mathcal{U}, \qquad \forall \tau \in [t, t + T_c], \tag{3.32}$$

$$\bar{\mathbf{u}}(\tau) = \bar{\mathbf{u}}(\tau + T_c), \qquad \forall \tau \in [t + T_c, t + T_p], \tag{3.33}$$

$$\bar{\mathbf{x}}(\tau) \in \mathcal{X}, \qquad \forall \tau \in [t, t + T_p], \tag{3.34}$$

where the T_p and T_c are the prediction and control horizon. The $\mathcal{U}$ is a set of feasible input values and $\mathcal{X}$ is a set of feasible states. The bar denotes internal controller variables and $\bar{\mathbf{x}}(\cdot)$ is the solution driven by the input $\bar{\mathbf{u}}(\cdot)$ with initial condition $\mathbf{x}(t)$. The function F is a cost function. A nonlinear model accurately describes the system's dynamics even in severe conditions because of the disturbances and aerodynamic forces that can be modelled as part of the model's dynamic [60]. Therefore, the NMPC perform nicely in agile manoeuvres at high-speed [61].

3.4 Comparison of LMPC and NMPC

The comparison of the LMPC and NMPC is not an easy task. NMPC does not assume a linearized operation point, and thus the optimisation problem solution is more precise, especially in agile manoeuvres. The nonlinear model and constraints should even better describe the system's limits. On the other hand, while solving the NMPC, some precautions must be made to ensure stability. The computation of the problem is also not a simple task and may need much more time to find a solution than a LMPC. The comparison of LMPC and NMPC for the UAV control is stated in [62], where the authors claim that LMPC had smaller Root Mean Square Error (RMSE) than the NMPC except the z-axis in the environment without the wind. Since the core of the toolkits, which define and solve the NMPC problem, uses the solvers for LMPC, this thesis will focus only on LMPC problem.

Chapter 4

Mathematical modelling of the UAV

This chapter is dedicated to modelling the quadrotor dynamics. The Equations of Motion are derived in the Sec. 4.2 using coordinate frames presented in Sec. 2.2. Since the derived model contains parameters that must be determined, the Sec. 4.3 is dedicated to the identification of the model's parameters. The last section Sec. 4.4 in this chapter is dedicated to the creation of the linear model.

4.1 Mathematical overview of a quadcopter

Publication [63] describes two approaches for obtaining a quadrotor model in 3 DOFs and in 6 DOFs, together with the linear approximations, rewritten into the state space representation. In [64], the authors combine Earth-fixed frame and body-fixed coordinate frame to obtain nonlinear equations using Newton-Euler method. On the other hand, in paper [65] the equations of motion were derived using Euler-Lagrange equations, furthermore, the derived equations use quaternions instead of Tait-Bryan angles. In study [66] Newton-Euler method was used together with Euler-Lagrange method which shows that both methods lead to the same result. Moreover, in [66], [67], the square of angular velocities is used as a control input for the individual rotors. On the contrary, in [63], [68], [69] another system description using upward force and body torque as the control input is presented.

4.2 The Quadcopter Equations of Motion

This section studies the dynamics of the drone and describes the UAV's behaviour with ODEs, firstly with kinematics and secondly with kinetics. Since the UAV is operating in 6 DOFs, the displacements and rotations are described by generalised positions

$$\boldsymbol{\eta} = [\boldsymbol{\eta}_p^\intercal, \boldsymbol{\eta}_o^\intercal]^\top = [x, y, z, \phi, \theta, \psi]^\top \in \mathbb{R}^3 \times \mathcal{S}^3, \tag{4.1}$$

and velocities

$$\boldsymbol{\upsilon} = [\boldsymbol{\upsilon}_l^\intercal, \boldsymbol{\upsilon}_a^\intercal]^\top = [v_x, v_y, v_z, v_\phi, v_\theta, v_\psi]^\top \in \mathbb{R}^6, \tag{4.2}$$

where vector $\boldsymbol{\eta}$ is composed of position $\boldsymbol{\eta}_p = [x, y, z]^\top$ and orientation $\boldsymbol{\eta}_o = [\phi, \theta, \psi]^\top$. Euler angles are described in intrinsic zyx-convention, where yaw and roll angles are defined on the interval $[-\pi, \pi]$ and pitch range is $[-\frac{\pi}{2}, \frac{\pi}{2}]$. The vector $\boldsymbol{\upsilon}$ includes linear velocity $\boldsymbol{\upsilon}_l = [v_x, v_y, v_z]^\top$ and Euler rate vector $\boldsymbol{\upsilon}_a = [v_\phi, v_\theta, v_\psi]^\top$. Moreover, the position $\boldsymbol{\eta}$ and velocity $\boldsymbol{\upsilon}$ are defined in the local ENU coordinate frame $\mathcal{W}$ (see Fig. 2.1), which simplifies the usage of the Equations of Motion in the MPC prediction steps. The velocities can also be described in the body-fixed coordinate frame as follows

$$\boldsymbol{\nu} = [\boldsymbol{\nu}_l^\intercal, \boldsymbol{\nu}_a^\intercal]^\top = [u, v, w, p, q, r]^\top \in \mathbb{R}^6, \tag{4.3}$$

where $\boldsymbol{\nu}_l = [u, v, w]^\intercal$ is linear velocity and $\boldsymbol{\nu}_a = [p, q, r]^\intercal$ denotes angular velocity. The detailed drawing of the UAV in the body-fixed coordinate frame can be seen in Fig. 4.1.

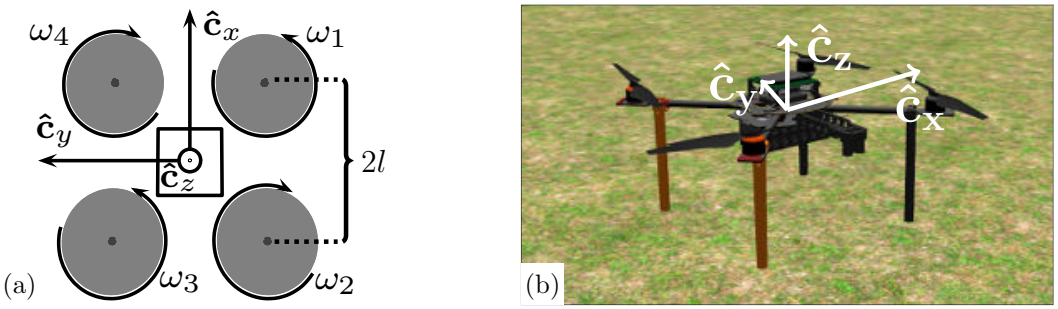


Figure 4.1: UAV body frame. (a) Top view with engine's rotations representation. (b) Coordinate image on a simulation model.

4.2.1 Nonlinear model

The nonlinear model of the UAV was derived in 6 DOFs. The position vector (4.1) and velocity vector (4.2) were defined in the local coordinate frame, and thus the kinematics part of the equations of motion is

$$\dot{\boldsymbol{\eta}} = \boldsymbol{\upsilon}. \tag{4.4}$$

The kinetics part of the equations of motion studies both the motions and their causes, and therefore, the equations can be derived either using Newton-Euler method or Euler-Lagrange approach. Both ways led to the same results and can be found in the articles that were cited at the beginning of Sec. 4.1. According to [66], it can be assumed that the local frame is inertial and therefore the centrifugal force

$$\mathbf{F}_c = \boldsymbol{\nu}_a \times (m\boldsymbol{\nu}_l) \tag{4.5}$$

can be neglected. The equation for the translational dynamics (2nd Newton's law) is defined like in [70] as

$$\ddot{\boldsymbol{\eta}}_p = -g\,\hat{\boldsymbol{e}}_3 + \mathbf{R}_c^w \frac{\mathbf{T_C}}{m}, \tag{4.6}$$

where g denotes gravitation acceleration, m is the quadcopter weight, $\mathbf{R}_c^w \in SO(3)$ is a rotation matrix that transforms from the copter's body-fixed frame to the ENU local reference frame and $\mathbf{T_C}$ is a total thrust (total force). The force is generated in the body-fixed frame as

$$\mathbf{T_C} = \begin{bmatrix} 0 \\ 0 \\ k \cdot \sum_{i=1}^{4} \omega_i^2 \end{bmatrix}, \tag{4.7}$$

where k is a lift constant for the motor with a propeller and ω is an angular velocity of the rotor (will be studied in further detail in the following section). To transform the (4.7) into

the local frame of reference, the matrix $\mathbf{R}_c^w$ is used. The $\mathbf{R}_c^w$ is an intrinsic *xyz*-convention Euler transformation matrix, which is equivalent to

$$\mathbf{R}_c^w = \underbrace{\begin{bmatrix}\cos(\psi) & -\sin(\psi) & 0\\ \sin(\psi) & \cos(\psi) & 0\\ 0 & 0 & 1\end{bmatrix}}_{\mathbf{R}_{z,\psi}} \cdot \underbrace{\begin{bmatrix}\cos(\theta) & 0 & \sin(\theta)\\ 0 & 1 & 0\\ -\sin(\theta) & 0 & \cos(\theta)\end{bmatrix}}_{\mathbf{R}_{y,\theta}} \cdot \underbrace{\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(\phi) & -\sin(\phi)\\ 0 & \sin(\phi) & \cos(\phi)\end{bmatrix}}_{\mathbf{R}_{x,\phi}}, \tag{4.8}$$

where $\mathbf{R}_{x,\phi} \in SO(3)$, $\mathbf{R}_{y,\theta} \in SO(3)$ and $\mathbf{R}_{z,\psi} \in SO(3)$ are rotation matrices which define individual rotations around axes $\hat{\mathbf{c}}_\mathbf{x}$, $\hat{\mathbf{c}}_\mathbf{y}$ and $\hat{\mathbf{c}}_\mathbf{z}$, accordingly in body-fixed frame.

The Euler angles acceleration is given by the Euler-Lagrange equation together with the transformation from the body-fixed coordinate frame to the local coordinate frame as

$$\ddot{\boldsymbol{\eta}}_o = \underbrace{(\mathbf{T}_c^w \mathbf{I}_B \mathbf{T}_w^c)}_{\mathbf{J}}{}^{-1} \left(\boldsymbol{\tau}_\boldsymbol{C} - \mathbf{C}_\mathbf{t}\dot{\boldsymbol{\eta}}_o\right), \tag{4.9}$$

where $\mathbf{C_t}$ is the Coriolis term, which contains centripetal and gyroscopic terms as is shown in [71]. The $\boldsymbol{\tau}_\boldsymbol{C}$ stands for total torque and has following form

$$\boldsymbol{\tau}_\boldsymbol{C} = \begin{bmatrix} l \cdot k \cdot (-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2)\\ l \cdot k \cdot (-\omega_1^2 - \omega_4^2 + \omega_2^2 + \omega_3^2)\\ \sum_{i=1}^4 b\omega_i^2 + \mathbf{I_M}\dot{\omega}_i \end{bmatrix}. \tag{4.10}$$

Assuming that the drone is symmetrical X-configuration, propellers are in one plane and the rotor's distance from the centre of mass is

$$l = d\cos(45°), \tag{4.11}$$

where d is equal to the length of the drone's arm. The b is the drag constant for the motor with propeller and $\mathbf{I_M}$ denotes the inertia moment of the motor. However, $\mathbf{I_M}$ can be left out due to its negligible effects on a quadcopter, the moment of inertia for the UAV's body (frame) can not be forgotten. The inertial matrix is $\mathbf{I}_B \in \mathbb{R}^{3x3}$ and denotes how much force is needed to start or stop rotating around the specific axis, where the off-diagonal terms are products of inertia and diagonal terms are moments of inertia. Since quadcopters are build symmetrically about both $\hat{\mathbf{c}}_\mathbf{x}$ and $\hat{\mathbf{c}}_\mathbf{y}$ axes, inertia matrix has the form

$$\mathbf{I}_B = \begin{bmatrix} I_{xx} & 0 & 0\\ 0 & I_{yy} & 0\\ 0 & 0 & I_{zz}\end{bmatrix}. \tag{4.12}$$

The transformation from the body-fixed frame to the local coordinate frame is done by the transformation matrix $\mathbf{T}_c^w$. According to [72], the transformation matrix $\mathbf{T}_w^c = (\mathbf{T}_c^w)^\top$ is defined as

$$\mathbf{T}_w^c = \mathbf{R}_{x,\phi}^\intercal \mathbf{R}_{y,\theta}^\intercal \mathbf{R}_{z,\psi}^\intercal \hat{\boldsymbol{e}}_\mathbf{3} + \mathbf{R}_{x,\phi}^\intercal \mathbf{R}_{y,\theta}^\intercal \hat{\boldsymbol{e}}_\mathbf{2} + \mathbf{R}_{x,\phi}^\intercal \hat{\boldsymbol{e}}_\mathbf{1} = \begin{bmatrix}1 & 0 & -\sin(\theta)\\ 0 & \cos(\phi) & \cos(\theta)\sin(\phi)\\ 0 & -\sin(\phi) & \cos(\theta)\cos(\phi)\end{bmatrix}. \tag{4.13}$$

Moreover, $\mathbf{T}_c^w$ is not defined for $\theta = \frac{\pi}{2} + k\pi, k \in \mathbb{Z}$. Fortunately, it is not expected that the UAV will do such a manoeuvre.

The nonlinear description of the drone can be expanded by the various forces which affect the behaviour of the UAV in flight. Such forces can be blade flapping, aerodynamic

drag, total thrust variation in translational flight and vortex [73]–[75]. However, all these extensions increase the complexity of the model, which increases the difficulty of identifying individual parameters. Moreover, in [76] with reference to Mahony et al. (2012) is mentioned that *"is widely shown in the literature that the control can achieve high performance with the simplified model of the rotor."*

The final Equations of motions (12 ODEs) are defined as follows

$$\dot{\boldsymbol{\eta}} = \boldsymbol{\upsilon}, \tag{4.14}$$

$$\ddot{\boldsymbol{\eta}}_p = -g\,\hat{\boldsymbol{e}}_3 + \mathbf{R}_c^w \frac{\mathbf{T_C}}{m}, \tag{4.15}$$

$$\ddot{\boldsymbol{\eta}}_o = \mathbf{J}^{-1}\left(\boldsymbol{\tau}_{\boldsymbol{C}} - \mathbf{C_t}\dot{\boldsymbol{\eta}}_o\right). \tag{4.16}$$

4.3 Model identification

In the previous section, the equations of motion were defined. Since the aforementioned equations contain parameters that can be either directly measured or calculated indirectly based on other measurements, this section is dedicated to finding the numerical values for these parameters. Methods for identification of variables can be found in the publications [65], [67], [77]. The UAV chosen for this thesis is built on a Tarot T650 frame with all the necessary electronics — computer, battery and sensors for detecting a USV. Furthermore, Sec. 2.5.1 is dedicated to a detailed description of the Hardware (HW) of the UAV.

The overall weight of the UAV differs for simulation and the real world and therefore the model's identification must be made separately for the simulation and real world. Since the simulation does not take into account the weight of the cover (which protects the drone against water) and with floats on the drone's legs, the drone's weight considered for the simulation is $m_{sim} = 3.5\,\text{kg}$ and for the real world is $m_{real} = 4.9\,\text{kg}$. Quadcopter's dimensions are stated in the table Table 4.1.

Symbol	Value	Description
d	0.325 m	length from the Center Of Gravity (COG) to the rotor shaft
r_b	0.25 m	body radius
h_b	0.15 m	body height

Table 4.1: Quadrotor's dimensions.

Moment of inertia

Moment Of Inertia (MOI) can be calculated using an assumption that the quad-copter body can be replaced with a solid cylinder of radius r_b and height h_b. To compute the MOI more precisely, a motor with a propeller can be used as a point mass at the end of the drone's arm. Since there is a difference between the simulation and real-world drone the computation must be done for both. Firstly, for the simulation, the calculation of the MOI is identical for

$$I_{xx_{sim}} = I_{yy_{sim}} = \underbrace{2(m_m + m_p)l^2}_{\text{point mass}} + \underbrace{(m_{sim} - (m_m + m_p))\frac{3r_b^2 + h_b^2}{12}}_{\text{perpendicular moment of inertia}} = 0.0761\,\text{kg}\,\text{m}^2, \tag{4.17}$$

due to the symmetrical shape of the UAV, where $m_m = 0.148$ kg stands for BLDC motor weight including cables and $m_p = 20.2$ g stands for carbon fiber propeller weight. The moment around the third axis (I_{zz}) is calculated as

$$I_{zz_{sim}} = \underbrace{4(m_m + m_p)l^2}_{\text{point mass}} + \underbrace{(m_{sim} - (m_m + m_p))\frac{r_b^2}{2}}_{\text{parallel moment of inertia}} = 0.1397\,\text{kg m}^2. \tag{4.18}$$

Secondly, the MOI must be computed for the real-world drone and therefore the results are stated in Table 4.2.

$I_{xx_{real}}$ (kg m^2)	$I_{yy_{real}}$ (kg m^2)	$I_{zz_{real}}$ (kg m^2)
0.1006	0.1006	0.1835

Table 4.2: Calculated moment of inertia for the real-world drone.

Another approach that can be used to obtain the MOI is called a Bifilar torsional method [78]–[80]. This method uses a Bifilar pendulum to determine a MOI by an experiment and therefore it can be used to clarify the results calculated above. Thanks to this method the resulting MOI should be much more precisely determined, because the calculation method assumes a homogeneous object, which the UAV is not. The equation which is used to calculate MOI using the Bifilar pendulum method is

$$I_{\text{real}} = \frac{mgd_s^2T^2}{16\pi^2 h_s}\ (\text{kg m}^2), \tag{4.19}$$

where m is mass of the measured object, g is the gravitational acceleration, d_s is the distance between the attachment of two strings, h_s is length of the strings and T is period of the measured movement. The results from this method are shown in the Table 4.3.

$I_{xx_{\text{real}}}$ (kg m^2)	$I_{yy_{\text{real}}}$ (kg m^2)	$I_{zz_{\text{real}}}$ (kg m^2)
0.1169	0.1218	0.1721

Table 4.3: Moment of inertia obtained from experimental Bifilar pendulum method.

These results clearly illustrate that the real-world UAV is not as symmetrical as was assumed in the calculation of the MOI. On the other hand, it must be taken into account that the measurements used in Bifilar pendulum were not made in laboratory conditions, which would ensure precise measurement of the pendulum period. Nevertheless, to decrease the measurement error the pendulum test was performed three times for each axis, which reduces the error caused be the human factor. Therefore it is possible to use these outcomes in real-world experiments instead of using calculated MOI.

Both methods have pros and cons, where each of these methods is used to determine the MOI for either real-world UAV or simulation UAV. MOI used in simulation and in real-world experiments are denoted in Table 4.4.

Symbol	Value	Description
$I_{xx_{sim}} = I_{yy_{sim}}$	0.1168 kg m^2	MOI around $\hat{\mathbf{c}}_\mathbf{x}$ and $\hat{\mathbf{c}}_\mathbf{y}$ for the simulated experiments
$I_{zz_{sim}}$	0.2204 kg m^2	MOI around $\hat{\mathbf{c}}_\mathbf{z}$ for the simulated experiments
$I_{xx_{real}}$	0.1169 kg m^2	MOI around $\hat{\mathbf{c}}_\mathbf{x}$ for the real-world experiments
$I_{yy_{real}}$	0.1218 kg m^2	MOI around $\hat{\mathbf{c}}_\mathbf{y}$ for the real-world experiments
$I_{zz_{real}}$	0.1721 kg m^2	MOI around $\hat{\mathbf{c}}_\mathbf{z}$ for the real-world experiments

Table 4.4: Resulted moment of inertia both for simulated and for real-world drone.

Propeller model

The propeller constants k and b are linear to to the ω^2, therefore, they can be obtained indirectly from measuring the *Tarot 4114 320KV* motor paired with the propeller *Tarot 1555* on a measuring device. The measuring stand was created to measure the thrust and the angular speed. The result from the measurement can be found on MRS github page in section *Hardware/Motor tests*[1]. Unfortunately, the stand used for this experiment was not capable of measuring the motors torque directly, therefore it had to be computed using the model

$$\tau_m = \frac{UI\epsilon}{2\pi \frac{n_{rpm}}{60}}, \tag{4.20}$$

where the U is measured voltage, I measured current, ϵ is an estimated efficiency (85 %) and n_{rpm} stands for the measured propeller's RPM. The motor constants can be calculated as

$$k = \frac{f_i}{\omega_i^2}, \tag{4.21}$$

$$b = \frac{\tau_{M_i}}{\omega_i^2}. \tag{4.22}$$

The resulted lift constant and drag constant are stated in Table 4.5.

Symbol	Value	Description
k	$6.6936 \cdot 10^{-5}$	lift constant
b	$1.9692 \cdot 10^{-6}$	drag constant

Table 4.5: Calculated propeller constants.

4.4 Linear approximation and discretization

In order to provide the linear approximation of the equations of motion which were derived in Sec. 4.2, an equilibrium point is needed [81]. Such a point could be hover, which is defined as a static position of the UAV with zero derivatives of position. Thus, both linear and angular speeds must be near zero, which also implies that the roll and pitch must be near zero as well. Position and yaw angle can be arbitrary, therefore the state vector looks as follows

$$\mathbf{x_h} = [\chi, \chi, \chi, 0, 0, \chi, 0, 0, 0, 0, 0, 0]^\top, \tag{4.23}$$

[1] https://ctu-mrs.github.io/docs/hardware/motor_tests.html

where $\chi \in \mathbb{R}$. The input vector

$$\mathbf{u_h} = [\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2]^\top \tag{4.24}$$

is composed of $\omega_i^2 \ldots i \in \{1, 2, 3, 4\}$, where each ω represents the angular speed of individual motors, from which the thrust can be calculated. Therefore, the overall sum of these four thrust figures must be equal to the drone's weight and also to ensure that the drone does not make any unwanted, potentially destabilising movements, these values must be identical. By linearizing and discretizing, using zero-order hold on the inputs, the aforementioned model in the operational point we get discrete LTI system.

Chapter 5

Trajectory planning

This chapter is dedicated to the methods for trajectory planning. Firstly, the reformulation of the MRS UAV control pipeline is shown in Sec. 5.1. The state automaton, which was developed to monitor the individual states of the UAV's flight, is introduced in Sec. 5.1.1. The trajectory generator itself was built on the MPC based full-state tracker mentioned in Sec. 3.2 and is responsible for creating the constraint and feasible trajectory for the UAV. Therefore, the methodologies mentioned in this chapter extend a classical usage of the MPC and ensure smoother and much more precise properties, not only for the landing on the marine vessel's deck but also for the phase of the approach and tracking.

5.1 The principle of creating trajectories

The modification of the MRS pipeline, which was mentioned in Fig. 2.4, is a necessary step in the process of developing the trajectory generator. The modified pipeline is depicted in Fig. 5.1. The new pipeline is extended with *Observation & position sharing* block, *Localisation*

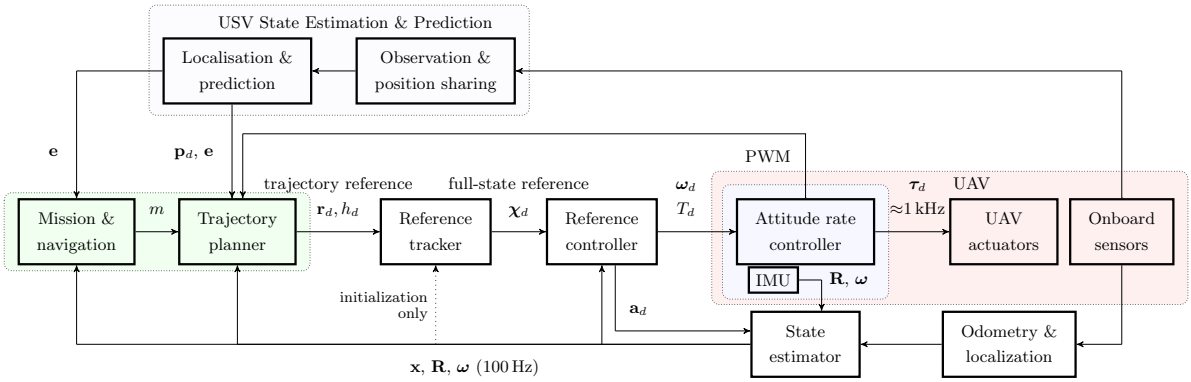


Figure 5.1: A modified diagram of the system architecture, which expands Fig. 2.4 with reformulation of the Mission & navigation.

& prediction block and finally with a *Trajectory planner* block. Furthermore, the *Mission & navigation* block is reformulated to contain the state machine responsible for determining flight mode (m). The estimation of the USV's and UAV's states is used as input. The information about the flight mode is shared with the *Trajectory planner*. The *Trajectory planner* also accepts the estimation ($\mathbf{e}$) of the USV's states, as well as the full-states prediction ($\mathbf{p}_d$) from the *Localisation & prediction* block. The embedded autopilot shares the information about the actual Pulse Width Modulation (PWM) signal command, which is sent to motors. Last but not least, the information about the UAV's current state ($\mathbf{x}, \mathbf{R}, \boldsymbol{\omega}$) is provided by the UAV's state estimator.

5.1.1 Mission & navigation

As was already mentioned above, the *Mission & navigation* block was modified, and therefore, this pipeline's stage is responsible for determining the mode in which the UAV is currently operating. The individual modes or states can be divided into three phases: APPROACH, FOLLOW and LANDING, as shown in Fig. 5.2. The state automaton starts in

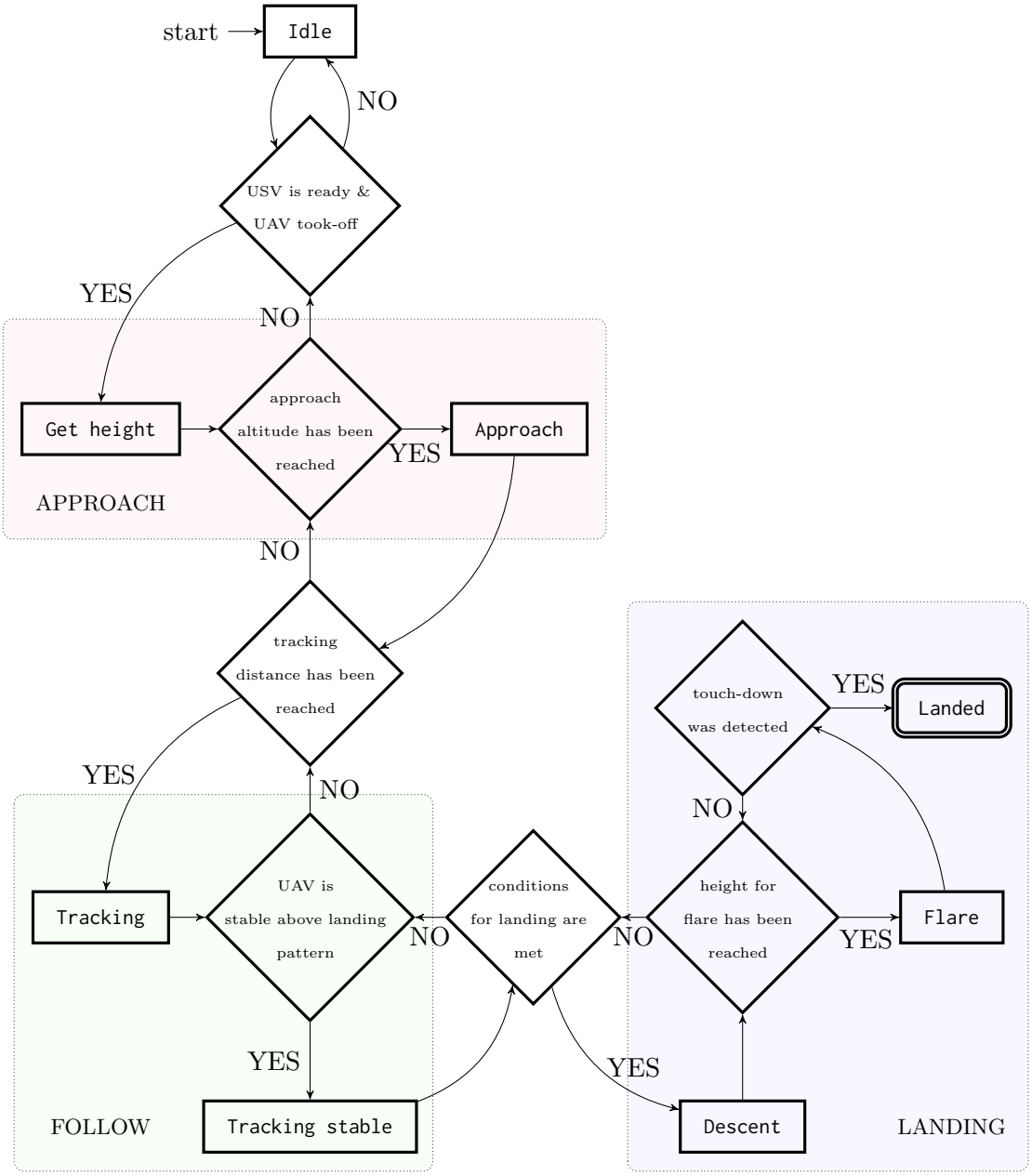


Figure 5.2: Mission & navigation state machine.

the state `Idle`, which awaits the USV's estimation and prediction. The UAV also waits until the end of the UAV's take-off manoeuvre. After the verification, the state machine passes into the state `Get height`.

State `Get height` is a member of the group, which denotes the APPROACH phase. The

position of the UAV after the state `Idle` can be very close to the water. Therefore, during the state `Get height`, the UAV reaches the desired approach altitude and modifies the yaw angle of the UAV to aim in the USV's direction. After that, the state automaton passes to the state `Approach`. This state denotes the stage in which the UAV approaches to the USV's x and y positions. The flight level is maintained, and the yaw angle is continuously updated to aim directly in the USV's position, throughout the approach stage. This reduces the threat of possible collisions with other vessels, which can be in the direction of flight while the UAV approaches the USV's position. Moreover, the yaw angle adjustment ensures that the USV is found in the FOV of the optical sensors at the earliest possible moment. In a moment when the UAV is not far from the USV, the state automaton changes phase from the APPROACH to the FOLLOW.

The FOLLOW phase is divided into two stages. The first state is called `Tracking` and denotes the final approach above the vessel's deck. The approach altitude is exchanged for tracking altitude. In a moment when the UAV is above the boat's deck stable the state automaton passes to the state `Tracking stable`. The UAV remains in this state until the conditions for landing are met, and then the state automaton passes into the LANDING phase.

The LANDING phase is the most challenging, hence, it is divided into three stages. The first stage is called `Descent`. It denotes an event in which the UAV starts descending towards the USV's deck and continues until the UAV's height reaches the distance from the boat's deck determined for the beginning of the `Flare` manoeuvre. This manoeuvre is quite similar to the manoeuvre that is commonly known from the aeroplane's final stage of the approach, just before the aircraft touches down on the runway. The procedure involves slowing down the vertical speed to smooth out the landing and reduce the impact of the wheels touching the ground. Since the boat's deck is rarely stationary, the predicted attitude angles of the USV's deck are taken into account during the `Flare` manoeuvre.

The touch-down is verified using an estimation of the needed thrust to maintain the flight. After touching down, the UAV's motors are turned off, and the state automaton passes to the state `Landed`.

5.1.2 Trajectory planner

The trajectory planner sometimes called a trajectory generator ensures the generation of a trajectory which is supplied to the MRS trajectory reference tracker. A graphical representation of the trajectory creation process is shown in Fig. 5.3. According to this figure, the actual UAV's mode, together with the predicted USV's states, are supplied to the *Full-state reference* block. Based on this information, the full-state reference ($\mathbf{s}_r$) is created and provided to the *LMPC optimisation*. The LMPC can also be modified based on the UAV's phase of flight. The penalisation matrices and hard constraints matrices can be dynamically adjusted according to the UAV's mode. For example, during the APPROACH phase, the penalisation matrix is changed to mainly penalise the position and yaw angle. During the TRACKING phase, it is appropriate to penalise the position, linear velocities and yaw angle. On the other hand, during the flare manoeuvre, the penalisation matrix is modified such as the roll and pitch angles are taken into account.

The output from the *LMPC optimisation* is conditioned by the used model, which was derived in Chap. 4. Thanks to that, a vector ($\mathbf{u}$) of the individual motor's angular speeds is obtained. This is then fed into a linear model from which the states of the UAV into the

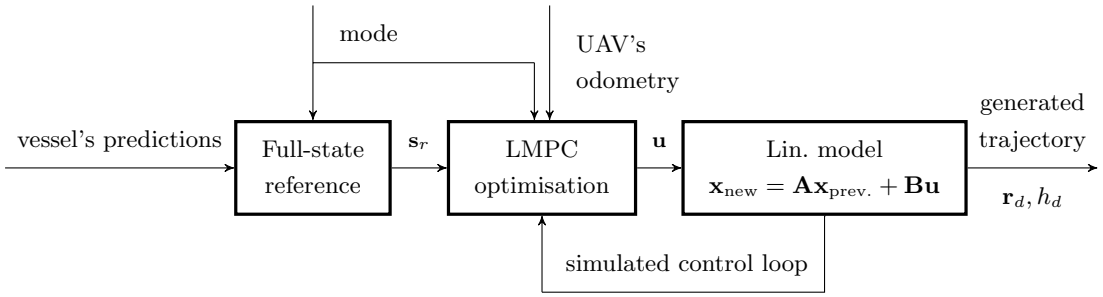


Figure 5.3: A diagram of the trajectory planner, where the "mode" is a state from the state automaton, $\mathbf{s}_r$ is a modified full-state reference that needs to be tracked. The $\mathbf{u}$ is a sequence of the outputs forwarded into the linear model. The linear model is responsible for creating the desired trajectory.

future are obtained and from which only the position and heading are supplied to the MRS trajectory reference tracker.

Phase APPROACH

The UAV's approach above the USV's position is possible thanks to the boat unit and its communication with the UAV [24]. The USV shares the GPS position, and therefore, the UAV knows roughly where to fly even if the USV is not in the FOV of the sensors. Thus the trajectory can be generated from the start, and it is not necessary to search the area to localise the USV. During the process of creating the trajectory, a point in the 3D Space or an estimated position of the USV can serve as the reference states.

It is also appropriate to determine the UAV's heading during the flight. The UAV is equipped with a stereo camera placed in the exact position. Therefore, to maximise the FOV of the camera, the UAV's front must be rotated in the direction in which the USV's position is estimated. The heading is calculated as

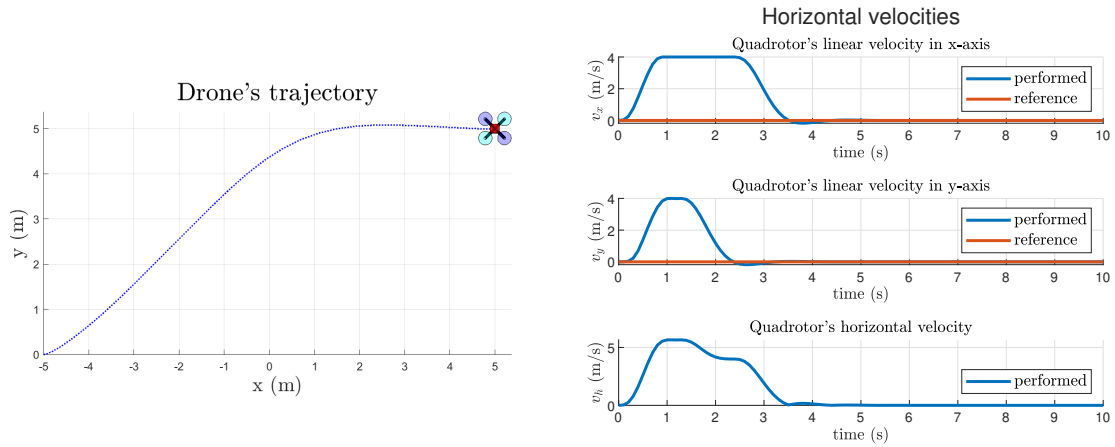
$$h = \text{atan2}(y_d - y_n, x_d - x_n), \tag{5.1}$$

where y_d and x_d are the y and x positions of the vessel and y_n and x_n are the estimated UAV's positions in individual axes.

The classical approach of the MPC is to minimise the tracking errors independently for the x-axis and the y-axis as is shown in Fig. 5.4. This can lead to the not optimal solution from the perspective of the length of the trajectory. Since the heading to the desired position is known, the calculation of the maximal speed in the individual horizontal axes can be calculated. This ensures that the resulting speed vector will point directly in the direction of the vessel's position and will limit the maximal horizontal speed such that the maximal velocity of the UAV will be in every horizontal direction the same. Therefore, the hard constraints must be modified dynamically based on the heading of the UAV. The overall horizontal speed is calculated as

$$v_h = \sqrt{v_x^2 + v_y^2}, \tag{5.2}$$

where v_x is a velocity parallel with the x-axis and v_y is parallel with y-axis in local ($\mathcal{W}$) coordinate frame. Since the horizontal speed is going to be limited, the velocity along the



(a) UAV's trajectory.

(b) UAV's horizontal velocities with maximal speed $7.07\,\mathrm{m\,s^{-1}}$.

Figure 5.4: x-axis and y-axis velocities are constrained independently.

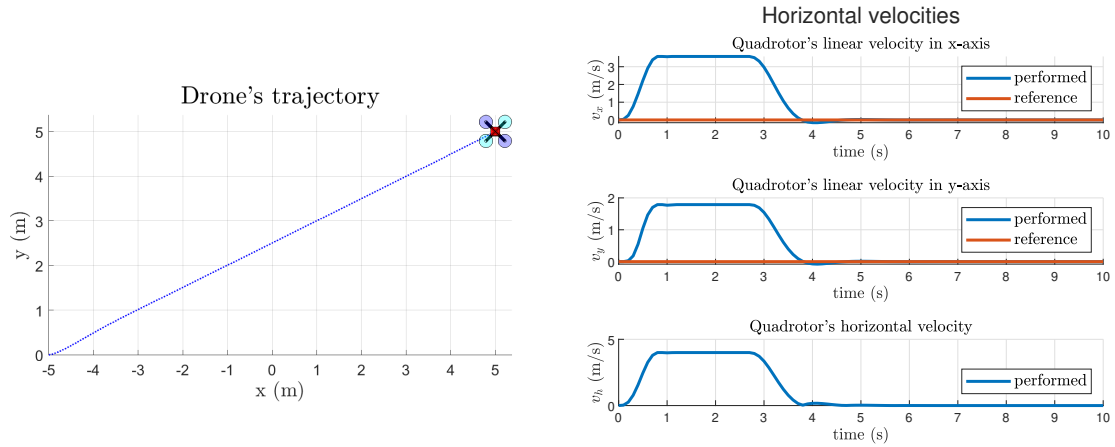
x-axis is

$$v_x = \cos(h) v_h \tag{5.3}$$

and the contribution of the velocity in the y-axis is

$$v_y = \sin(h) v_h. \tag{5.4}$$

The ideal solution for tracking distant paths or points and minimising the length of the UAV's trajectory is shown in Fig. 5.5. However, it should be noted that it is not an optimal solution in terms of time.



(a) UAV's optimal trajectory.

(b) UAV's orizontal velocities.

Figure 5.5: The x-axis and y-axis velocities boundaries are based on a full velocity vector. The time to reach the desired position is slower since the x-axis velocity was reduced.

Phase FOLLOW

The USV is followed by the UAV using the vessel's predicted trajectory and linear velocities in the horizontal plane. This combination ensures smooth tracking even at the boat's speed, which achieves its maxima $4\,\mathrm{m\,s^{-1}}$. The UAV's heading is also controlled to ensure that the vessel will be in the FOV of the sensors. During tracking, the UAV continuously monitors the incoming prediction of the USV's future states. These states must be within the USV's limits so that they can be tracked. The covariance of the predicted states is also monitored to be within a predefined range.

Phase LANDING

The landing phase includes the descent, flare and touch-down manoeuvres. Firstly, the descent manoeuvre provides the final approach above the vessel's deck. Since the landing is the most dangerous phase, the UAV's position deviation from the vessel's centre in the 2D plane is strictly monitored. In the case of a greater deviation, the landing is aborted to prevent falling the UAV into the water. Since MPC was used for generating trajectory, the operating space for planning can be limited using the dynamically changing hard constraints on position. This ensures that the trajectory planner can not generate a trajectory into the water. In case of UAV misses the landing platform, the trajectory generator ensures that the UAV will stop at a safe distance above the water. The reference trajectory is created, so the z-position is sampled to maintain the specific descent speed. This approach is more beneficial during the flare manoeuvre.

By specifying the vertical approach velocity and not just the descent speed, the UAV lands on the vessel's platform with a predefined velocity regardless of the USV's vertical motions. The illustration of the landing manoeuvre is shown in Fig. 5.6. Since the USV is not moving linearly only, the tilt of the vessel's deck must be taken into account. The information about the vessel's roll and pitch is supplied to the reference tracker to create the trajectory, which will align the UAV's attitude with the USV's during the landing.

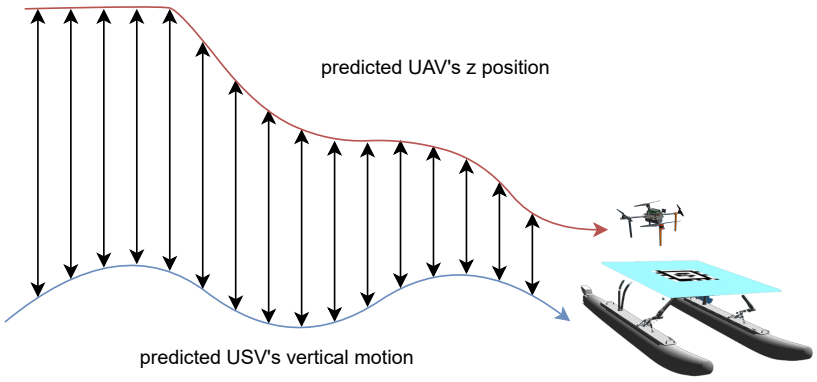


Figure 5.6: The representation of the descent manoeuvre.

Chapter 6

Development tools

The tools which were used to design and verify the proposed solution in this thesis are described in this chapter. The description of the simulation environment in Sec. 6.3 makes up majority of this chapter. Another important topic is about optimisation solvers and tool-kits described in Sec. 6.4.

6.1 Robot Operating System

The Robot Operating System (ROS) is an integral part of the MRS UAV control system [82]. The control pipeline was displayed in the previous chapters — block schemes interconnected with arrows showing that data is exchanged there. And this is precisely for what the ROS was made.

The ROS implements a peer-to-peer network of processes to share the information synchronously or asynchronously using ROS communication infrastructure. The process, which performs some computation, is in the ROS language called a node, e.g., a node responsible for estimating the UAV's actual state. However, the node can not be assigned without the master. Without the ROS Master, the nodes would not be able to find each other and communication between them could not be established. Asynchronous communication is done by streaming the data over topics using the publisher on one side and the subscriber on the other. The data are packed into structures called messages. On the other hand, synchronous communication uses request and replay interactions between two nodes via services.

Packages, which are the main unit for organising codes, are defined on the ROS Filesystem level. The code may be written in MATLAB, C++ or Python. The ROS Package also includes the definition of the messages and services data structures. Since the ROS has many distributions nowadays, it is necessary to mention that the Noetic distribution was used.

6.2 MATLAB

MATLAB is a powerful programming and numeric computing platform which was used to design and verify elementary tasks in this thesis. In MATLAB was described the UAV's nonlinear model, proposed in Sec. 4, which was then linearized and discretized. The resulting LTI model was used in the definition of the UAV in C++ code. Moreover, the MATLAB version 2021a already supports a ROS and also custom ROS messages, and therefore, the first simplified tests were implemented using MATLAB directly. These tests included the standalone simulations verifying the QP solvers and their settings. Then the interconnection with MRS simulation environment was made. MATLAB was also used to gather the results from the simulations and real-world experiments to examine the proposed trajectory generator.

6.3 Simulation environment

Before testing the proposed solution in the real world, it is always good to validate and verify the results in some simulator. Therefore, the simulation environment in the Gazebo[1] simulator was created for this purpose. The Gazebo simulator is a very popular physics simulator with a realistic environment, which natively supports ROS. Thanks to its popularity, many robotics groups create plugins, models and environments [83]–[86]. And this turned out to be a great advantage. The simulation's environment core was inspired by the project called Virtual RobotX (VRX) [87], which was created for the maritime VRX Competition. The GitHub vrx repository[2] was merged with MRS Gazebo-based simulations[3] to obtain the simulation environment with the USV and UAV located on the offshore.

The screenshot with the running simulation and its graphical interface can be seen in Fig. 6.1. The ROS 3D visualisation, tmux session and dynamic reconfigurator are also shown in the same picture.

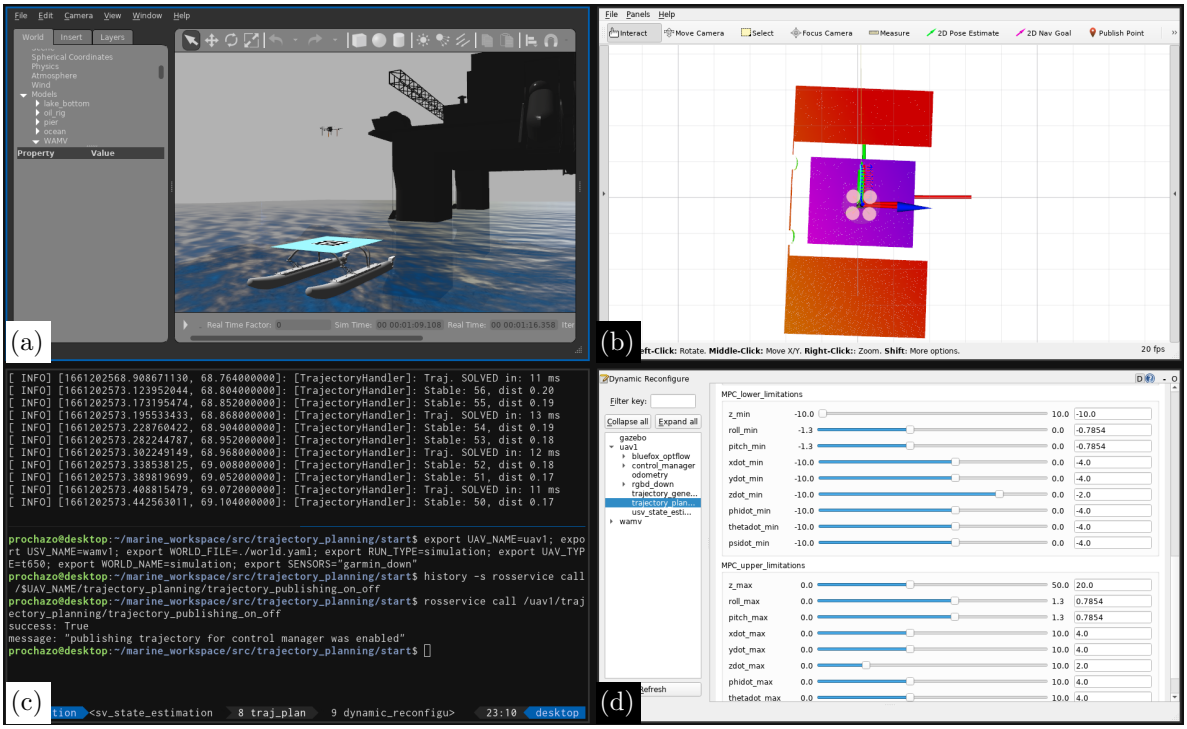


Figure 6.1: Screenshot of the running simulation. (a) Gazebo 3D simulation. (b) ROS visualisation (Rviz) 3D visualisation. (c) Tmux terminal. (d) Dynamic reconfigurator.

6.3.1 Water and Wave-field modelling

Modelling the water and its motion is a key aspect of the simulations. The closer the simulated environment is to reality, the better and safer the tests are in the real world. Therefore, the Gazebo plugins were developed to model the ocean waves and simulate their

[1] https://gazebosim.org
[2] https://github.com/osrf/vrx
[3] https://github.com/ctu-mrs/simulation

physical behaviour [87]. Water is modelled as a liquid which forces on the object according to Archimedes' principle

$$F_B = -\rho V g, \tag{6.1}$$

where ρ is the density of the medium, V is the object's volume, which is submerged in the water, and g is the gravitational acceleration. The water's drag force

$$F_d = \beta_l m \frac{V}{V_T}(\boldsymbol{v_w} - \boldsymbol{v_c}) \tag{6.2}$$

is modelled to prevent oscillations. β_l is the linear drag coefficient, m is the mass of the object, V_T is the total volume of the object, $\boldsymbol{v_w}$ is the velocity of the fluid current and $\boldsymbol{v_c}$ is the velocity of the object. To eliminate angular velocity oscillations, the drag torque is modelled as

$$T_d = \beta_a m \frac{V}{V_T} L^2 \omega, \tag{6.3}$$

where β_a is the angular drag coefficient, L is the average width of the object and ω is the angular velocity of the object in a body-fixed frame.

According to [87], water motion is modelled using the summation of Gerstner waves, representing the water surface as a trochoidal shape [88]. The horizontal and vertical displacements of the wave-field to the undisturbed horizontal location $\mathbf{x}_0 = [x_0, y_0]^\top$ with a vertical height of $\zeta_0 = 0$ are defined as

$$\mathbf{x}(\mathbf{x}_0, t) = \mathbf{x}_0 - \sum_{i=1}^{N} q_i (\boldsymbol{k}_i / k_i) A_i \sin(\boldsymbol{k}_i \mathbf{x}_0 - \omega_i t + \phi_i), \tag{6.4}$$

$$\zeta(\mathbf{x}_0, t) = \sum_{i=1}^{N} A_i \cos(\boldsymbol{k}_i \mathbf{x}_0 - \omega_i t + \phi_i), \tag{6.5}$$

where each component wave is defined with steepness (q_i), amplitude (A_i), angular frequency (ω_i), random phase (ϕ_i) and wavevector ($\boldsymbol{k}_i$) in the direction of travel with the magnitude k_i. The N is the number of wave components at time t.

The component wave characteristic can be determined empirically or more robustly by sampling parametric wave spectrum $S_B(\omega)$. The second method ensures that the simulated waves are very close to the realistic open ocean waves. According to [87], the two-parameter (peak frequency ω_p and significant wave height H_s) Pierson-Moskowitz wave spectrum was used to obtain the wave-field [89]. The resulting expression for the wave-field is

$$S_B(\omega) = \left(\frac{\bar{H}_s}{H_s}\right)^2 \frac{\alpha g^2}{\omega^5} \exp\left(-\frac{5}{4}\left(\frac{\omega_p}{\omega}\right)^4\right), \tag{6.6}$$

where ω is an angular frequency, $\alpha = 8.1 \times 10^{-3}$ and $\bar{H}_s$ denotes independent significant wave height. The amplitude of the particular wave can be obtained using the equation

$$A_i^2 \approx 2 S(\omega_i) \Delta_{\omega_i}, \tag{6.7}$$

where ω_i are the sample locations along the spectrum and $\Delta\omega_i$ is the width of the frequency band associated with the individual samples. Moreover, the direction of the wave travel can be taken into account by using the following equation

$$E(\omega, \theta) = S(\omega) D(\omega, \theta), \tag{6.8}$$

where θ represents the wave direction and $D(\omega, \theta)$ is the directional spreading function [90].

The VRX wave plugin enables setting up the peak period $T_p = \frac{2\pi}{\omega_p}$, steepness q, mean wave direction vector, and wave height gain expressed as $K_H = \frac{\bar{H}_s}{H_s}$. Therefore, the height of the waves can be simulated. According to Table 6.1, the most represented height of the waves is between 1.25 m – 2.5 m. The wave's size is also associated with a wind [20, p. 13]. The wind's intensity, which corresponds with the wave's size described as a moderate sea, is a fresh breeze (8.0 m s^{-1}–10.7 m s^{-1}).

		Percentage probability %		
Description of sea	Wave height observed (m)	World wide	North Atlantic	Northern Nort Atlantic
Calm (glassy)	0			
Calm (rippled)	0–0.1	11.2486	8.3103	6.0616
Smooth (wavelets)	0.1–0.5			
Slight	0.5–1.25	31.6851	28.1996	21.5683
Moderate	**1.25-2.5**	**40.1944**	**42.0273**	**40.9915**
Rough	2.5–4.0	12.8005	15.4435	21.2383
Very rough	4.0–6.0	3.0253	4.2938	7.0101
High	6.0–9.0	0.9263	1.4968	2.6931
Very high	9.0–14.0	0.1190	0.2263	0.4346
Phenomenal	Over 14.0	0.0009	0.0016	0.0035

Table 6.1: Definition of sea state (SS) codes (Price and Bishop, 1974). Notice that the percentage probability for SS codes 0, 1 and 2 is summarised. Adopted from [27, p. 200].

6.3.2 Multirotor Unmanned Aerial Vehicle

The multirotor Unmanned Aerial Vehicle (UAV) model for the simulation is based on the Tarot t650 frame, which was used for real-life experiments. The UAV model copies and simulates the equipped electronic from the real-life drone mentioned in Sec. 2.5.1 including necessary sensors. In Fig. 6.2 are depicted the screenshots from the Gazebo simulator.

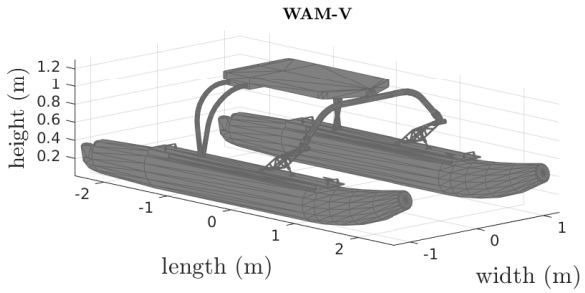
(a) The screenshot of the UAV.

(b) The screenshot of the UAV from the bottom showing the placement of the sensors.

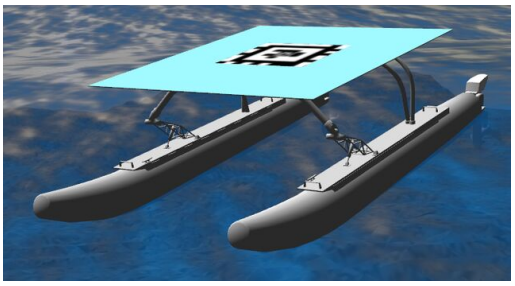
Figure 6.2: The UAV build on Tarot t650 frame used in Gazebo simulator.

6.3.3 Marine Unmanned Surface Vehicle

The USV was modelled as Wave Adaptive Modular Vessel (WAM-V), which is a special type of watercraft. WAM-V uses a system of suspensions connected to pontoons that reduce the movement created by waves [91]. Unfortunately, this improvement was not created in the simulation, and the USV was modelled as a rigid body which only looks like WAM-V. The vessel's model with the dimensions is shown in Fig. 6.3(a). The complete catamaran has an engine with a propeller at the back of each float. The landing platform is 2.4 m wide and 3 m long. On the top of the platform are situated UVDAR LEDs and AprilTag.



(a) The marine vessel's dimensions used in simulation.



(b) The screenshot of the USV from the Gazebo simulator.

Figure 6.3: (a) WAM-V model and (b) the screenshot from the simulator.

Vessel's propulsion system

As was mentioned above, the USV has two motors which are located on the vessel's stern. To control these motors in simulation, the Gazebo plugin was created according to [87]. The plugin specifies the thruster's type, characteristics, and position on the vessel. Moreover, the plugin ensures that the engines rotate around the joint at the attachment point. The maximal angle to angle of rotation is $\pm\pi$. Modelling of the thrust force can be linear or even nonlinear using Generalised Logistic Functions (GLFs). The thrust force is limited between $-100\,\mathrm{N} - 250\,\mathrm{N}$. Therefore, the maximal USV's velocity is $(4\,\mathrm{m\,s^{-1}})$.

To speed up the tests, the keyboard control[4] of the thrusters was added also according to [87]. The control key inputs are depicted in the Fig. 6.4, where the left diagram shows how the thrust can be controlled, for example, the left arrow marked as "left" uses the differential thrust of the two motors to make a vessel's turn. On the right side are shown keys that are used to control the motor's angle and the keys which are used to change the maximal or minimal velocities.

Path following

To prepare the simulation for the automated testing of the UAV's landing on the moving vessel, the USV controller was developed. The control was inspired by the conventional navigation system used in aviation, where the aircraft's path leads over the beacons. In our case, the beacons can be exchanged for virtual buoys that are placed on the water. Therefore,

[4] http://wiki.ros.org/teleop_twist_keyboard

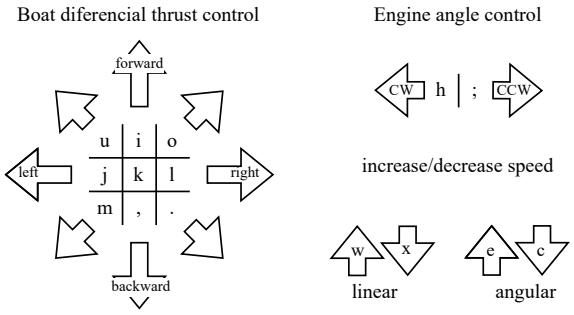


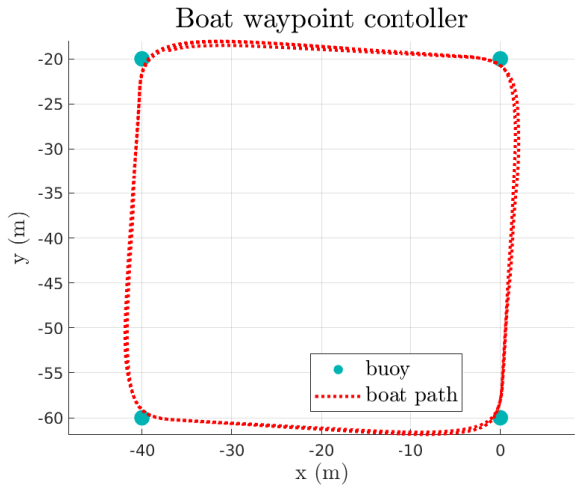
Figure 6.4: Keys for controlling the vessel using the keyboard.

the vessel's path is created from the set of waypoints, which are one by one visited in the determined order. In this case, the heading value of the waypoints is left out.

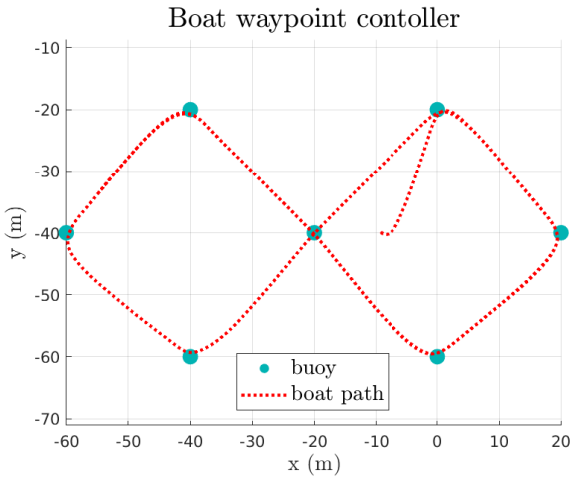
The USV's controller ensures that the boat's heading is set and that the motion vector points directly to the next waypoint. The heading towards the goal waypoint is calculated as

$$h_b = \text{atan2}(y_w - y_b, x_w - x_b), \tag{6.9}$$

where the subscript w denotes the actual waypoint and subscript b denotes boat. Since the current vessel's heading is known, the proportional component of the classical PID controller [92] is used to manage the motors' angle. The controller does not change the thrust of the motors. After reaching the current waypoint, the next waypoint is set as a goal (this can run in an infinite loop). The close loop system runs at 10 Hz and the user can tune the P-controller using dynamic reconfiguration. The user can also define the positions and order of the waypoints. The two illustrations of the USV's trajectory together with boys are shown in Fig. 6.5.



(a) Waypoints ordered in shape of square.



(b) Waypoint ordered in shape of figure-eight.

Figure 6.5: Waypoints in square order Fig. 6.5a and waypoints ordered in figure-eight Fig. 6.5b.

6.4 Optimisation Solvers and Tool-kits

Since the usage of MPC expanded in recent years, the number of solvers and even tool-kits for solving linear and nonlinear MPC problems have increased too. Choosing a proper solver is crucial for this thesis, and therefore, a few solvers were studied and investigated to determine the most suitable one. According to [93], the three types of optimisation strategies {Active-set methods, Interior-point methods, First-order methods} for solving the QP exist. The following section describes the QP solvers in a nutshell and chooses the proper one for this thesis.

Active-set methods, also known as the projection methods, were derived from the simplex algorithm [94] and were the first methods for solving QPs. The algorithm has two phases. The first phase tries to find the feasible point that satisfies all constraints without taking into account the objective function. The second phase iteratively lowers the objective function updating the active constraints. The disadvantage of this method is that the complexity grows exponentially with the number of constraints. The implementation of active-set methods can be found in solvers like MOSEK [95], GUROBI [96] and qpOASES [97]–[100]. Nowadays, when the MPC is very popular, we must also mention the tool-kits like Automatic Control and Dynamic Optimization (ACADO) [101] and its successor acados [102], [103].

Interior-point methods, also known as barrier methods, model the optimisation problem as a parameterised penalty function. The optimum is achieved by iterating the unconstrained optimisation problem with a varying barrier function. Contrary to the active-set methods, this method runs in polynomial time and solves more complex linear programming problems. The drawback of these methods is that they are not easily scaled for large problems. However, they are still used in commercial algorithms like MATLAB's quadprog [104], [105], OOQP [106] and CVXGEN [107]. It should be noted that the CVXGEN is currently used in the MRS trajectory tracker mentioned in Sec. 2.4.3. CVXGEN is a code generator for embedded convex optimisation that can be used to solve quadratic programs at high speed in the order of milliseconds. It is also very stable and robust for tasks up to hundreds of optimisation variables, and therefore, it is suitable for tasks which are required to be computed in real-time. There can also be pointed out the "connection" to this thesis because SpaceX uses the same optimiser to land with Falcon 9's first stage of the space rocket on the drone ship "Of Course I Still Love You" [108].

First-order methods use only the first derivative of the cost function and iteratively compute an optimal solution. Such algorithms can be Douglas–Rachford splitting [109] or Alternating Direction Method of Multipliers (ADMM) [110]. The advantage of the ADMM algorithm is its fast convergence with a relatively small number of iterations, which are computationally very cheap. Therefore, the solution is obtained fast, even for large-scale optimisation problems. On the other hand, the algorithm does not provide high-accuracy solutions. Another problem is that the first-order methods are not capable of detecting infeasibility. Hence the Operator Splitting Quadratic Program (OSQP) solver was proposed in [93]. The solver uses ADMM algorithm and can provide high-accuracy solutions using solution polishing. The primal and dual infeasibility detection was also resolved.

According to numerical examples shown in [93], the OSQP solver is very competitive to the qpOASES, GUROBI and MOSEK. To verify the results from the [93], additional experiments with qpOASES, quadprog and OSQP were done. The description and results of these tests are shown in Appx. A. Nevertheless, these tests show that the OSQP solver is most suitable for this thesis.

6.4.1 Operator Splitting Quadratic Program

Operator Splitting Quadratic Program (OSQP) [93] is one of the many solvers which can be used to solve the quadratic cost function with the linear constraints in a form

$$\text{mimize} \quad \frac{1}{2}\mathbf{x}^\intercal \mathbf{P}\mathbf{x} + \mathbf{q}^\intercal \mathbf{x}, \tag{6.10}$$

$$\text{subject to} \quad \mathbf{l} \leq \mathbf{A}\mathbf{x} \leq \mathbf{u}, \tag{6.11}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the decision variable, $\mathbf{P} \in \mathbb{S}_+^n$ denotes positive semi-definite matrix of the objective function with a vector $\mathbf{q} \in \mathbb{R}^n$. The constraints are defined by matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and restriction vectors $\mathbf{l} \in \mathbb{R}^m \cup \{-\infty\}^m$, which denotes the lower bounds and $\mathbf{u} \cup \{-\infty\}^m$, which denotes the upper bounds. The solver runs the ADMM algorithm which in every iteration produces a tuple $(\mathbf{x}_{[k]}, \mathbf{y}_{[k]}, \mathbf{z}_{[k]})$ with $\mathbf{x}_{[k]} \in \mathbb{R}^n$ and $\mathbf{y}_{[k]}, \mathbf{z}_{[k]} \in \mathbb{R}^m$ and auxiliary variables $\tilde{\mathbf{x}} \in \mathbb{R}^n$ and $\tilde{\mathbf{z}} \in \mathbb{R}^m$. The pseudo-code is shown in Alg. 1, where $k \in \mathbb{Z}^+$ denotes an iteration step, $\sigma > 0$ and $\rho > 0$ are the step-size parameters, $\alpha \in (0, 2)$ is the relaxation parameter, and Π denotes the Euclidean projection onto $\mathcal{C}$, which is closed and convex set defined with the boundaries $\mathbf{l}$ and $\mathbf{u}$. $\boldsymbol{\nu} \in \mathbb{R}^m$ is the Lagrange multiplier associated with constraint $\mathbf{A}\mathbf{x} = \mathbf{z}$.

Algorithm 1 OSQP solver algorithm

Require: initial values $\mathbf{x}_0, \mathbf{z}_0, \mathbf{y}_0$ and parameters $\rho > 0, \sigma > 0, \alpha \in (0, 2)$

1: **while** termination criteria is not satisfied **do**

2: $(\tilde{\mathbf{x}}_{[k+1]}, \boldsymbol{\nu}_{[k+1]}) \leftarrow$ solve linear system $\begin{bmatrix} \mathbf{P} + \sigma\mathbf{I} & \mathbf{A}^\intercal \\ \mathbf{A} & -\rho^{-1}\mathbf{I} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_{[k+1]} \\ \boldsymbol{\nu}_{[k+1]} \end{bmatrix} = \begin{bmatrix} \sigma\mathbf{x}_{[k]} - \mathbf{q} \\ \mathbf{z}_{[k]} - \rho^{-1}\mathbf{y}_{[k]} \end{bmatrix}$

3: $\tilde{\mathbf{z}}_{[k+1]} \leftarrow \mathbf{z}_{[k]} + \rho^{-1}(\boldsymbol{\nu}_{[k+1]} - \mathbf{y}_{[k]})$

4: $\mathbf{x}_{[k+1]} \leftarrow \alpha\tilde{\mathbf{x}}_{[k+1]} + (1 - \alpha)\mathbf{x}_{[k]}$

5: $\mathbf{z}_{[k+1]} \leftarrow \Pi(\alpha\tilde{\mathbf{z}}_{[k+1]} + (1 - \alpha)\mathbf{z}_{[k]} + \rho^{-1}\mathbf{y}_{[k]})$

6: $\mathbf{y}_{[k+1]} \leftarrow \mathbf{y}_{[k]} + \rho(\alpha\tilde{\mathbf{z}}_{[k+1]} + (1 - \alpha)\mathbf{z}_{[k]} - \mathbf{z}_{[k+1]})$

7: **end while**

The algorithm's convergence is checked by the termination criteria based on the specified tolerance of the primal and dual residual's norm of the optimisation problem. If the problem is feasible, the primal and dual residuals in every iteration step of the algorithm are calculated as

$$r_{\text{prim}[k]} = \mathbf{A}\mathbf{x}_{[k]} - \mathbf{z}_{[k]}, \tag{6.12}$$

$$r_{\text{dual}[k]} = \mathbf{P}\mathbf{x}_{[k]} + \mathbf{q} + \mathbf{A}^\intercal \mathbf{y}_{[k]}, \tag{6.13}$$

where $\mathbf{z} \in \mathbb{R}^m$ is an additional decision variable and $\mathbf{y} \in \mathbb{R}^m$ is the Lagrange multiplier associated with the constraint $\mathbf{A}\mathbf{x} = \mathbf{z}$ and $\mathbf{y} \in N_{\mathcal{C}}(z)$ denotes the normal cone of $\mathcal{C}$ at $\mathbf{z}$. The termination criteria for stopping the algorithm are defined as

$$||\mathbf{r}_{\text{prim}[k]}||_\infty \leq \epsilon_{\text{prim}}, \tag{6.14}$$

$$||\mathbf{r}_{\text{dual}[k]}||_\infty \leq \epsilon_{\text{dual}}, \tag{6.15}$$

where the tolerance level is set as

$$\epsilon_{\text{prim}} = \epsilon_{\text{abs}} + \epsilon_{\text{real}} \max\{||\mathbf{A}\mathbf{x}_{[k]}||_\infty, ||\mathbf{z}_{[k]}||_\infty\}, \tag{6.16}$$

$$\epsilon_{\text{dual}} = \epsilon_{\text{abs}} + \epsilon_{\text{real}} \max\{||\mathbf{P}\mathbf{x}_{[k]}||_\infty, ||\mathbf{A}^\intercal \mathbf{y}_{[k]}||_\infty, ||\mathbf{q}||_\infty\}. \tag{6.17}$$

The absolute tolerance ϵ_{abs} and relative tolerance ϵ_{rel} are positive numbers definable by the user, where the default value is set to $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} = 10^{-3}$ and for high accuracy is recommended the number $\epsilon_{\text{abs}} = \epsilon_{\text{rel}} 10^{-5}$. It may happen that the solution will be infeasible [111]. Therefore, the OSQP check for these situations and generates the certificate of primal or dual infeasibility.

Polishing is an additional algorithm used to obtain the high-accuracy solution from the final ADMM iterates. According to [93], the polishing procedure is done by solving an additional system of linear equations

$$\begin{bmatrix} \mathbf{P} + \delta\mathbf{I} & \mathbf{A}_{\mathcal{L}}^{\mathsf{T}} & \mathbf{A}_{\mathcal{U}}^{\mathsf{T}} \\ \mathbf{A}_{\mathcal{L}} & -\delta\mathbf{I} & \\ \mathbf{A}_{\mathcal{U}} & & -\delta\mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}}_{\mathcal{L}} \\ \hat{\mathbf{y}}_{\mathcal{U}} \end{bmatrix} = \begin{bmatrix} -\mathbf{q} \\ \mathbf{l}_{\mathcal{L}} \\ \mathbf{u}_{\mathcal{U}} \end{bmatrix} \tag{6.18}$$

where $\mathcal{L}$ and $\mathcal{U}$ are the active sets of constraints and $\delta > 0$ is a regularisation parameter with value $\delta \approx 10^{-6}$. The linear system is always solvable, but by regularisation, the solution includes a small error. To reduce this error, the iterative refinement procedure is applied [112].

Since the OSQP is an iterative method, the time duration for solving the optimisation problem arises with a number of iterations. Therefore, the OSQP solver ensures a warm start to reduce the number of iterations. It is also possible to reduce the maximal amount of iteration and also define the check termination interval. The time duration can be limited by setting the run-time limit. Other parameters that can be tuned are shown in OSQP documentation [113].

It is also worth mentioning that the aforementioned solver can be called from C, C++, Python, MATLAB and more. The OSQP source code is available online at github[5]. Since the solver itself is written in C language, the additional library osqp-eigen[6] was used.

[5] https://github.com/osqp/osqp
[6] https://github.com/robotology/osqp-eigen

Chapter 7

Simulations and verifications

In order to verify the proposed solution of the trajectory generator for landing on the marine vessel's deck described in Chap. 5, a few analyses were performed. This chapter describes these experiments and shows their results. Most of the experiments were performed in Gazebo simulation environment mentioned in Sec. 6.3, but some preliminary tests were made in MATLAB. For example, the information on how to set up the OSQP solver was obtained from the analysis done in Matlab. Graphs showing the behaviour of the solvers are shown in Appx. A.4 and display the prediction horizon size change versus duration time and RMSE. Based on these results, the length of the prediction horizon and also values of penalisation matrices were chosen. The settings of the MPC based trajectory generator are stated in Table 7.1. It should be noted that penalisation matrices are dynamically modified, and their values change based on the actual UAV's state. Limitations of attitude angles, linear velocities and angular rates are declared in the same table. According to this adjustment, a trajectory reference tracker mentioned in Sec. 2.4.3 was set up.

Symbol	Value	Description
N	20	number of prediction horizon steps
$\mathbf{Q}$	$[30, 30, 40, 1, 1, 50, 1, 1, 1, 1, 1, 1]$	diagonal objective function matrix
$\mathbf{P}$	$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$	diagonal terminal penalty weight matrix
$\mathbf{R}$	$[0.1, 0.1, 0.1, 0.1]$	diagonal objective function matrix
$\phi,\ \theta$	$\pm 0.7854\,\mathrm{rad}$	roll and pitch min/max angle
$v_x,\ v_y$	$\pm 4\,\mathrm{m\,s^{-1}}$	horizontal min/max velocity in x and y axis
v_z	$\pm 2\,\mathrm{m\,s^{-1}}$	vertical min/max velocity
$v_\phi,\ v_\theta$	$\pm 2\,\mathrm{rad\,s^{-1}}$	roll and pitch rotation min/max velocity
v_ψ	$\pm 1\,\mathrm{rad\,s^{-1}}$	yaw rotation min/max velocity

Table 7.1: Trajectory planner settings.

The trajectory generator is capable of generating trajectory from take-off to touchdown, therefore, additional parameters which define altitude or approach speed are stated in Table 7.2. The following sections focus on individual experiments that validate the proposed

Symbol	Value	Description
h_a	$15\,\mathrm{m}$	approach height
h_t	$7\,\mathrm{m}$	tracking height
v_{la}	$1\,\mathrm{m\,s^{-1}}$	landing approach velocity
v_{fa}	$0.5\,\mathrm{m\,s^{-1}}$	flare approach velocity

Table 7.2: Trajectory parameters.

trajectory generator in the Gazebo simulator. For every test it is common that waves are different in amplitude or period. Thanks to the path-following controller developed in Sec. 6.3.3

the experiments can be divided into two groups — based on whether the vessel is using the engine and following the predefined path or not.

7.1 Marine vessel carried by a current

The first experiments investigate the trajectory planning used for approach and landing on the USV which is influenced by waves. The waves affect the USV's movements not only heaving but also moving sideways. Therefore, the boat does not remain in one place and continuously moves in the direction of a current. The size of the heave movement depends on a wave's peak amplitude and period. Both parameters were set up in the Gazebo simulation environment in different combinations to test the proposed trajectory generator together with MRS UAV control in various situations.

7.1.1 Landing on a vessel on a moderate sea

Landing with a UAV on a boat's deck in calm water is not as challenging as landing on a vessel, which is located on a moderate sea. According to the Table 6.1, the most probable waves have a height between 1.25 m and 2.5 m. Thus the wave's peak amplitude was set to 0.5 m and period to 5 s to achieve these waves' proportions in a simulation. The resulting wave's height in the simulation varied between 0.6 m and 2.8 m. To discuss the results from these experiments, one of the most interesting tests was chosen to show results in the following graphs.

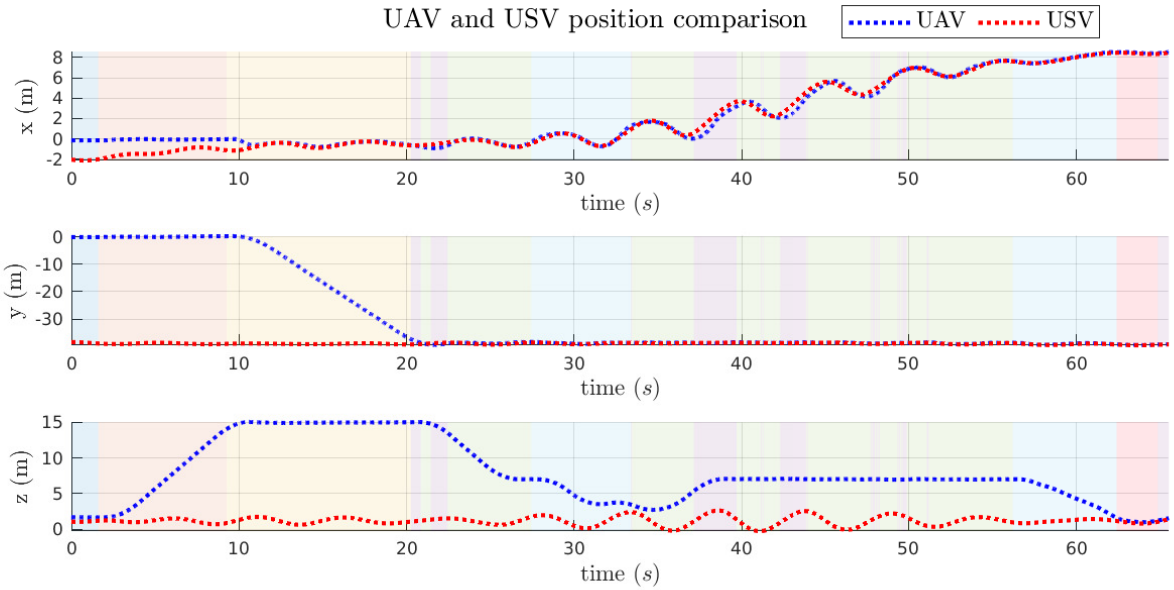


Figure 7.1: The comparison of the UAV's and USV's position. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

This experiment started with an UAV and an USV located 40 meters away from each other. The position of the UAV and USV throughout the experiment is shown in Fig. 7.1. In

the beginning, the UAV was prepared to follow the trajectory, but the trajectory generation was not turned on yet. This state is called Idle and lasted 1.6 s. After the trajectory generator was turned on, the UAV ascended to a height of 15 meters and adjusted a yaw angle to aim towards the vessel (see Fig. 7.3a). Then the UAV started approaching the USV's location maintaining an approach height and heading. The UAV was able to achieve USV's coordinates within 11 seconds at a speed of $4\,\mathrm{m\,s^{-1}}$ as is shown in Fig. 7.3b. The tracking of the USV

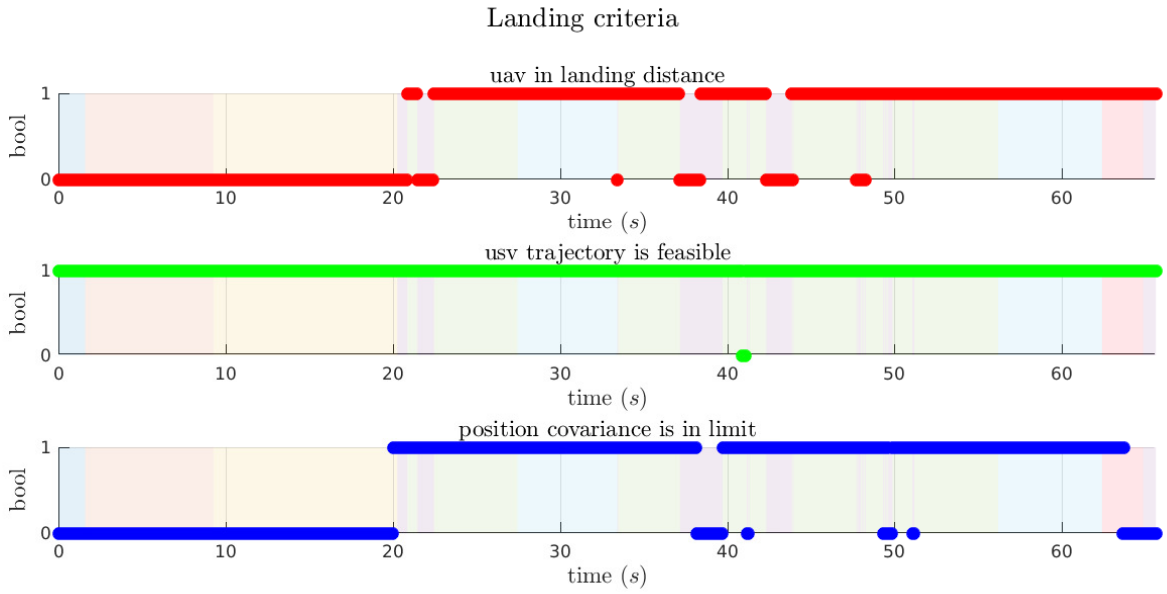
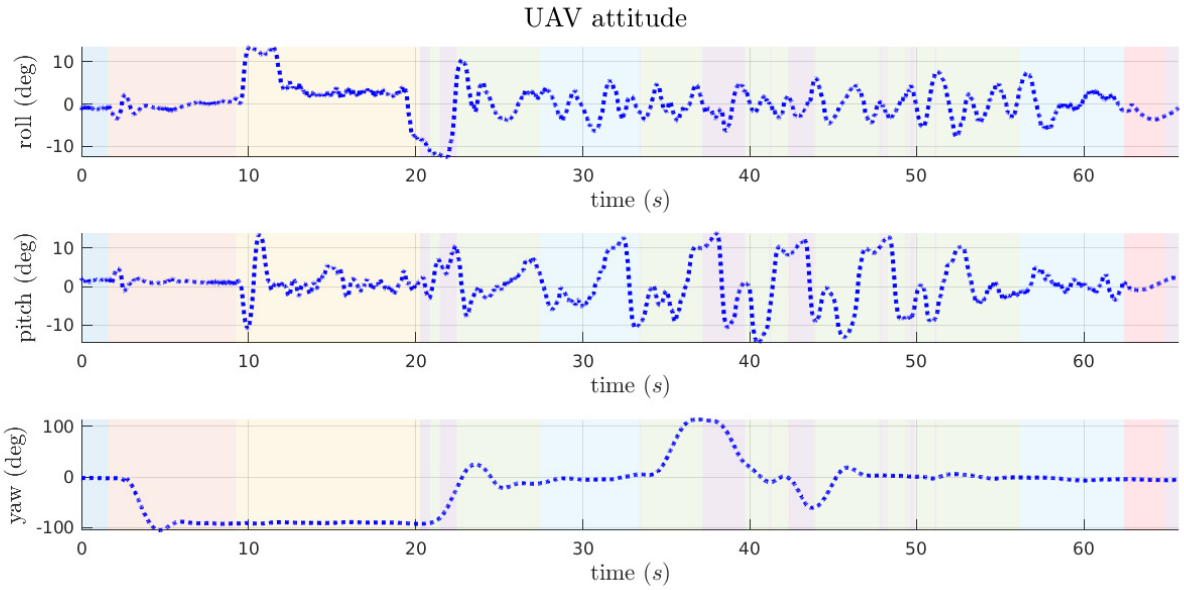


Figure 7.2: Landing criteria showing in which time which criterion was fulfilled. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

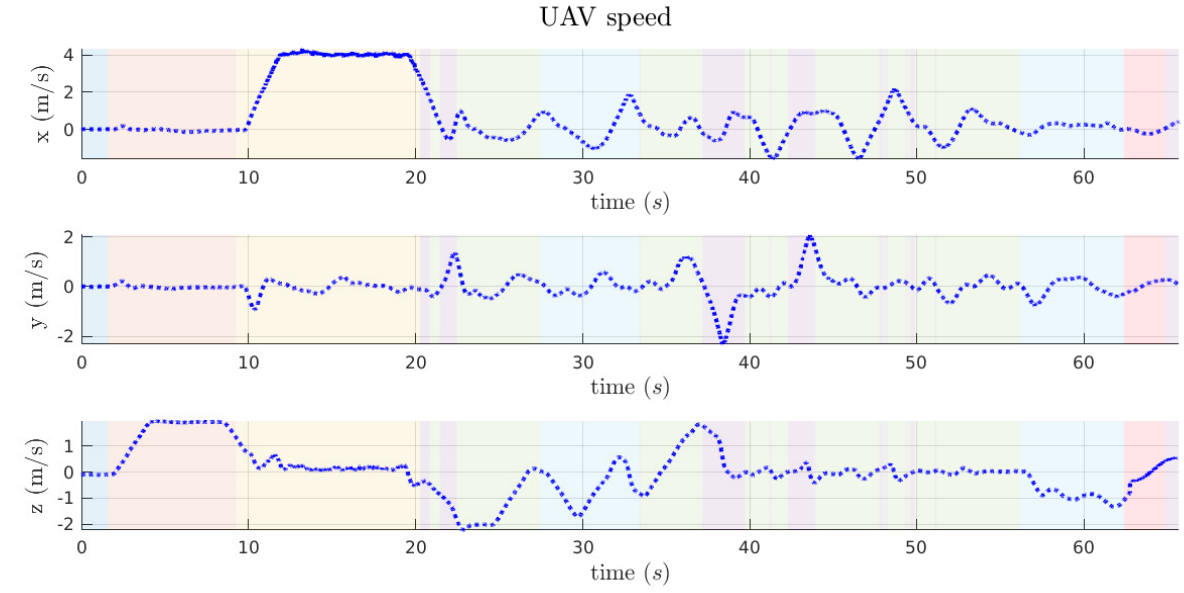
began at time $t = 20.3\,\mathrm{s}$ when the UAV started getting information from an AprilTag detector. This event is illustrated in Fig. 7.2 when the estimated position covariance decreased under a predefined limit. The UAV descended into a tracking flight level and aligned the yaw angle with the USV. The landing phase started at time $t = 27.5\,\mathrm{s}$ when landing criteria were met. Nevertheless, the size of the waves increased during the final descent, and the USV was forced to move more aggressively. This caused a deviation of the UAV from the vessel's deck which violated a landing distance criterion, and therefore, the landing phase was terminated at time $t = 33.5\,\mathrm{s}$. The landing error criterion was set to 0.5 m from the centre of the landing platform. After returning to the tracking altitude and aligning the yaw angle, the UAV waited monitoring a vessel's movement until time $t = 56\,\mathrm{s}$. While the UAV was tracking the vessel, the predicted trajectory was detected to be unfeasible by the UAV due to an excessive vertical velocity.

The final landing occurred without problems, and a flare manoeuvre began at time $t = 62.4\,\mathrm{s}$ and lasted just 300 ms before the UAV touched down on the vessel's deck. Measured approaching speed was $0.5\,\mathrm{m\,s^{-1}}$. Roll angle deviation was −1.5°, pitch angle deviation −0.3° and yaw angle deviation −0.1°. The deviation of the position was 19 cm in x-axis and −3 cm in y-axis . The aforementioned deviations are related to the USV's precise position and orientation obtained directly from the simulator. Hence, the estimation error can be one of the reasons for these deviations. The screenshot of the landed UAV after this test is shown in Fig. 7.4. This picture shows that even though the position error was not negligible, the

UAV did not land outside of the AprilTag (a square with a side length of 1 meter). Additional graphs from this experiment can be found in Appx. B.1.

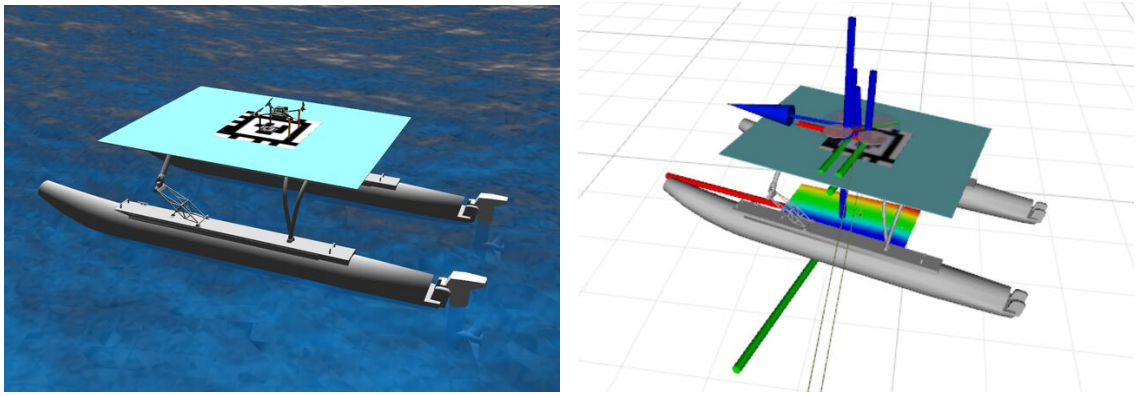


(a) UAV's attitude angles.



(b) UAV's linear velocities in the body-fixed coordinate frame.

Figure 7.3: UAV's odometry. Horizontal and vertical velocities Fig. 7.3b and attitude angles Fig. 7.3a. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare



(a) The screenshot from the Gazebo simulator. (b) The screenshot from the Rviz.

Figure 7.4: The snapshots from the simulation experiment after the touch-down.

7.1.2 Evaluation

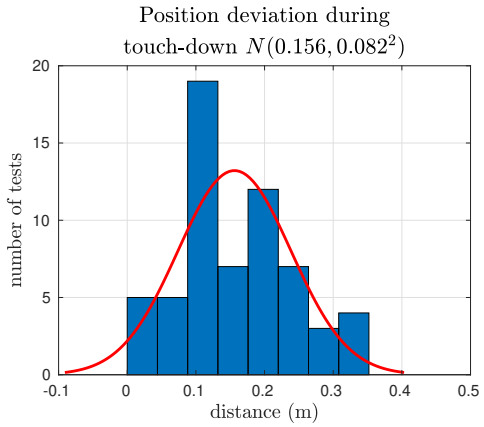
To verify the proposed trajectory generator, 62 tests of landings on a moderate sea were performed. The height, amplitude and frequency of waves were set up based on wave's simulation settings mentioned in Table 7.3. Settings for a landing manoeuvre are also denoted in the same table.

Symbol	Value	Description
T_p	$3\,\mathrm{s}, 4\,\mathrm{s}, 5\,\mathrm{s}$	wave's peak period
A	$0.2\,\mathrm{m}, 0.3\,\mathrm{m}, 0.4\,\mathrm{m}, 0.5\,\mathrm{m}, 0.6\,\mathrm{m}$	wave's amplitude
v_{la}	$1\,\mathrm{m\,s^{-1}}$	landing approach velocity
v_{fa}	$0.5\,\mathrm{m\,s^{-1}}$	flare approach velocity

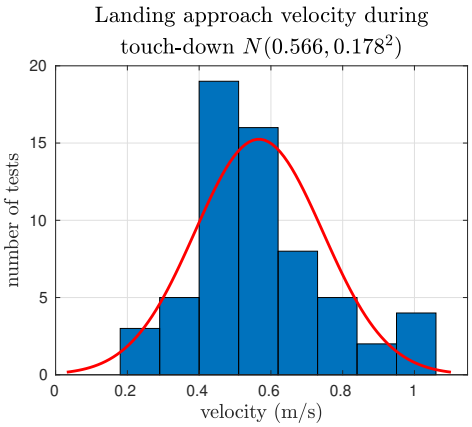
Table 7.3: Simulation settings for landing on a vessel carried by a current.

During these experiments, the boat was carried by the water flow, so roll, pitch and yaw values changed during individual tests based on a wave's field intensity, direction and amplitude. The waves' height was varying from 30 cm to 2.8 m. Roll and pitch values were in a range $\pm 10°$. Another critical factor for landing is the velocity in the horizontal direction or even more in the vertical. Since the boat was rocking due to the effects of waves, the vertical velocity was $\pm 1.5\,\mathrm{m\,s^{-1}}$ and the horizontal velocity $\pm 1\,\mathrm{m\,s^{-1}}$. All performed landings were examined, and the results are shown in Fig. 7.5.

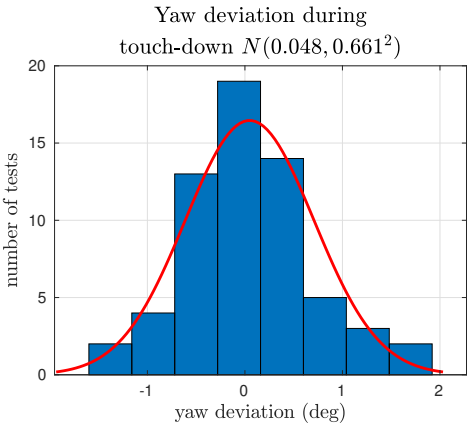
An essential factor for evaluating the trajectory planner for the landing on the USV's deck is a position error for touch-down. Therefore, graph Fig. 7.5a evaluates the position deviation of the UAV from the centre of the USV' deck. The mean value of an error was 15.6 cm. Moreover, it can be seen from the graph that most of the touch-downs (19) were performed about 10 cm far from the centre of the USV's deck. This means that the UAV did not land outside the area of the AprilTag. The observed landing error is apparently caused by the fact that during the flare manoeuvre, the UAV does not strictly minimise only the position deviation but also takes into account the attitude angles and velocities. It is also worth mentioning that the position deviation was measured from the USV's true position and not from the estimated position. Therefore, an estimation error may also play a role in the size of the deviation.



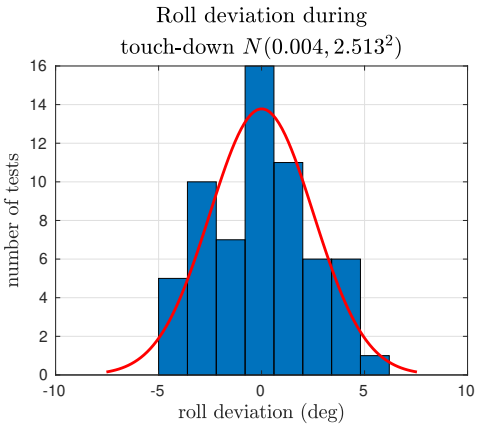
(a) UAV's position deviation from USV's centre during touch-down.



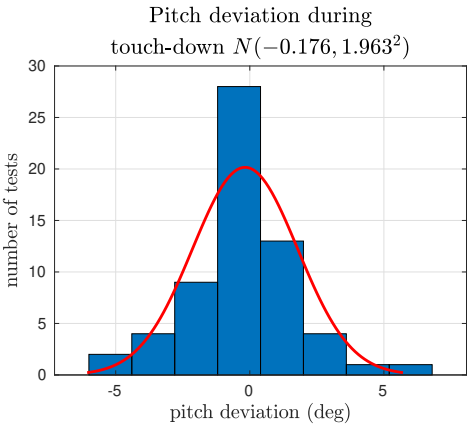
(b) UAV's approach speed during touch-down.



(c) UAV's yaw deviation from USV's during touch-down.



(d) UAV's roll deviation from USV's during touch-down.



(e) UAV's pitch deviation from USV's during touch-down.

Figure 7.5: Landing on a vessel on a moderate sea verification.

Another important aspect of the landing is the vertical approach speed during the flare manoeuvre. The results from tests are plotted in Fig. 7.5b, which depicts the normal distribution together with the histogram of the landing approach speeds. The landing approach speed was computed by subtracting the UAV descent speed and USV's vertical speed. According to this graph, the mean value of the measured approach speeds is $0.566\,\mathrm{m\,s^{-1}}$, which is close to the predefined value v_{fa}. On the other hand, a few landings performed touch-down with an approach speed equal to $1\,\mathrm{m\,s^{-1}}$. This could be caused by the wrong prediction of the waves' movement or by the acceleration/deceleration limitations of the trajectory tracker.

The remaining three graphs (Fig. 7.5d, Fig. 7.5e and Fig. 7.5c) depict roll, pitch and yaw angles deviations during the touch-down. The graphs are shown together with the normal distribution functions, where can be seen that the yaw angle deviation has a mean value close to zero. This means that during the landing, the UAV was aligned with the USV almost precisely with a standard deviation equal to 0.661°. Roll and pitch angle deviations had higher standard deviations during tests, but nonetheless, the results are satisfying.

Summary

It was shown in the above experiments that the trajectory generator is capable of generating complete trajectories from take-off to touch-down and that the landing is performed with a reasonable amount of harshness even on a moderate sea. Despite that, the overall system is not perfect yet.

During these experiments it was observed, that the accuracy of the estimation and prediction of the USV depends on a wave's field period. The lower the waves' period is, the less reliable estimation and prediction of the USV's states is. This is most likely caused by a fast change in a waves' height. Therefore, waves with a higher amplitude do not present as big of an issue as the smaller more erratic waves.

One of the trajectory generator's shortcomings that could be observed is an absence of accelerations in the UAV's model. This causes a discrepancy between the generated trajectory and MRS UAV trajectory tracker. Finally, yaw angle overshoot was observed during the experiments.

7.2 Marine vessel following a predefined path

The trajectory planner was developed to be universal and capable of generating a trajectory for landing on a moving vessel. A moving vessel is, in this case, the vessel which follows the path's waypoints (buoys) and performs desired changes of locations. The horizontal velocity was set to be $2\,\mathrm{m\,s^{-1}}$ on the non-wavy "glassy" sea. In a wavy environment, the vessel's velocity varies based on the waves' height, period and direction. The heaving frequency is changed based on the direction of the waves and the direction of the vessel's movement. Therefore, the paths were created so that the USV was moving in different directions during individual experiments. The path in a shape of a square ensures that the USV is moving lengthwise along the x and y axes. On the other hand, the path in a shape of a figure-eight ensures that the USV's movement is diagonal to the x and y axes and includes many turns.

7.2.1 Landing on a path-following vessel

The experiment started similarly to the test mentioned in Sec. 7.1.1, with the difference that the vessel was assigned to follow the waypoints with a predefined velocity. The waypoints were placed in a shape of a square with a side length of 80 meters. In the second phase, an approach altitude of 15 meters was ascended by the UAV, as well as the yaw angle was adjusted to aim towards the USV (see Fig. 7.7a). This manoeuvre was performed between time $t = 1.6\,\mathrm{s}$ and $t = 9.15\,\mathrm{s}$, after that, the UAV started to approach the moving vessel. According to Fig. 7.7b, the approach horizontal velocity was roughly $5.6\,\mathrm{m\,s^{-1}}$. It should be noted that even though the dynamic change of the velocity constraints was introduced in Sec. 5.1.2, it was not functioning correctly due to the absence of soft constraints. Therefore, this system was not used. The UAV caught up with the vessel at time $t = 32.2\,\mathrm{s}$ when the

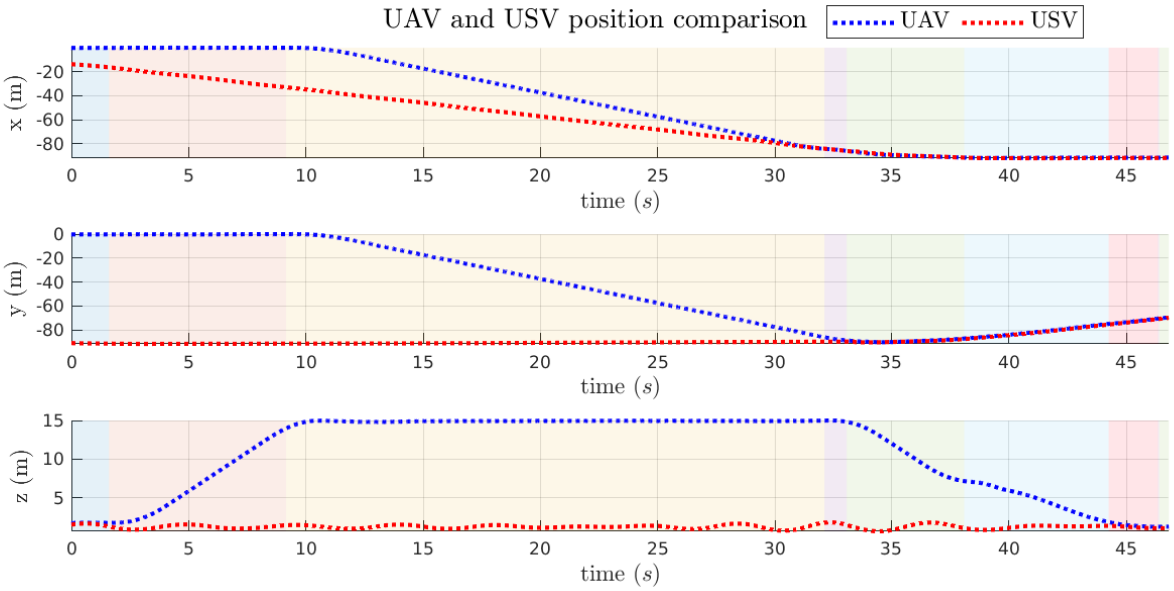


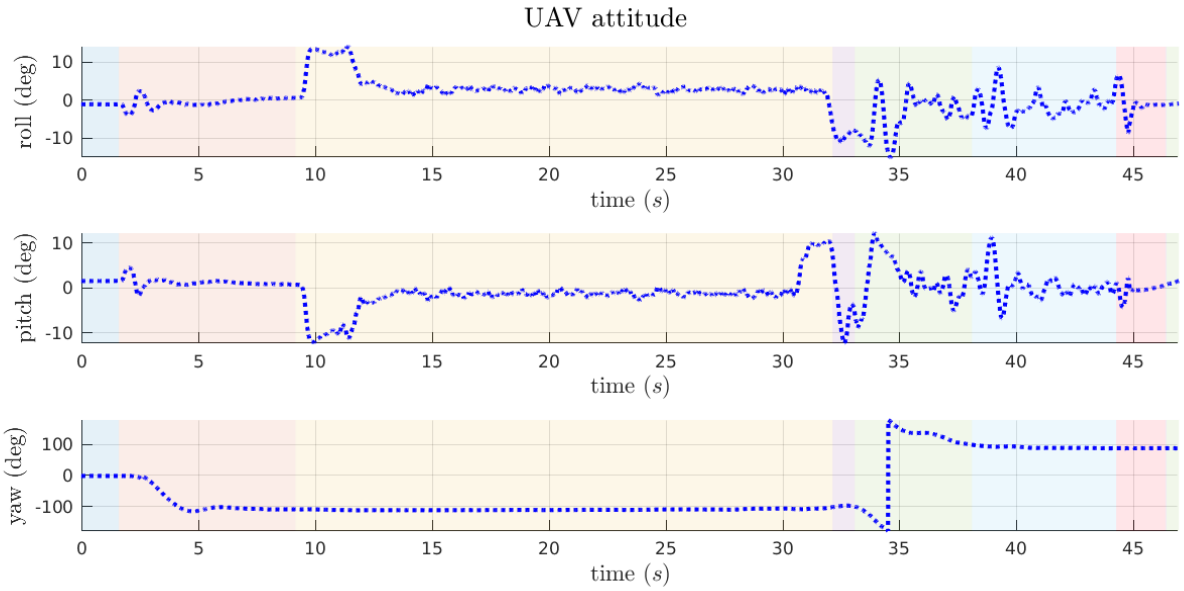
Figure 7.6: The comparison of the UAV's and USV's position. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

USV passed one of the waypoints and changed the direction of movement as can be seen in Fig. 7.6 which depicts the position of the UAV and the USV throughout the experiment. The attitude angles and linear velocities of the vessel can be found in the figures that are enclosed in Appx. B.2.

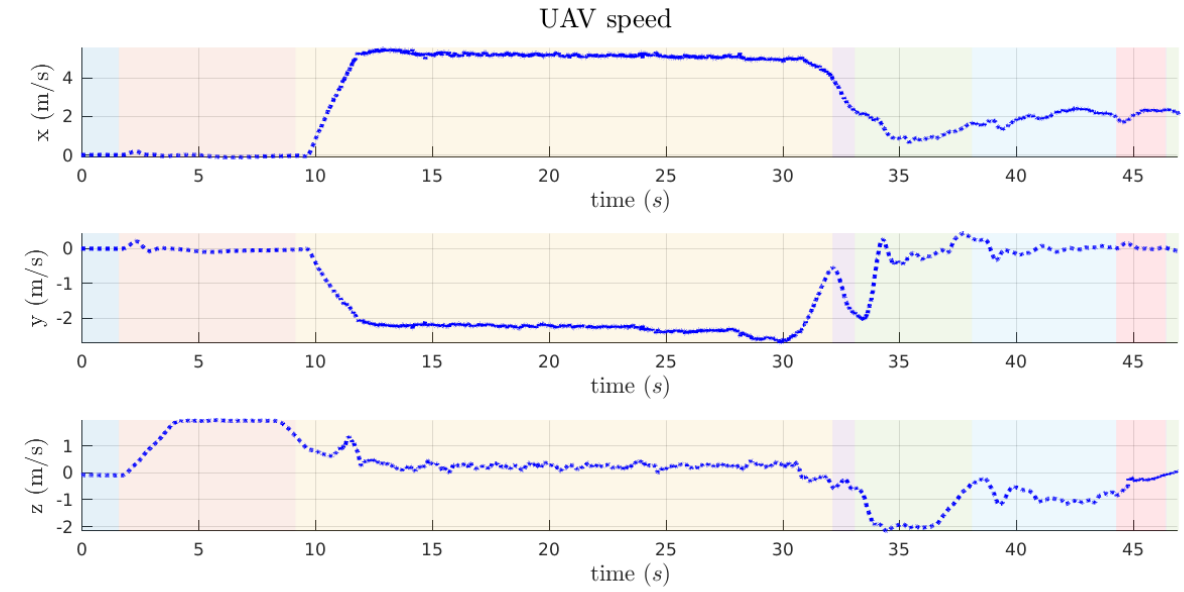
After an approach phase, the UAV started to track the USV which was accelerating in the direction of the y-axis and decelerating in the direction of the x-axis. Despite the current acceleration, landing criteria were met (see Fig. 7.8) and the UAV decided to perform the landing manoeuvre at time $t = 38.1\,\mathrm{s}$. The USV was still accelerating during the landing, but the UAV was able to fly in a stable way above the vessel's deck. The flare procedure began at time $t = 44.3\,\mathrm{s}$ and lasted 400 ms.

The touch-down was performed with an approach speed equal to $0.4\,\mathrm{m\,s^{-1}}$ with a position error 0.54 m away from the vessel's deck centre. The deviation of the roll angle was 7.67°, the pitch angle −1.26° and the yaw angle 0.38°. The speed at which the USV was moving

during the UAV's touch-down was $0.05\,\mathrm{m\,s^{-1}}$ in along the x-axis and $2.09\,\mathrm{m\,s^{-1}}$ along the y-axis. Since the UAV was aligned with the USV, the UAV's x-axis velocity in the body-fixed frame should be the same as the USV's y-axis in the global coordinate frame. Nevertheless, the reported velocity of the UAV was $1.8\,\mathrm{m\,s^{-1}}$.



(a) UAV's attitude angles.



(b) UAV's linear velocities in the body-fixed coordinate frame.

Figure 7.7: UAV's odometry. Attitude angles are shown in Fig. 7.7a, and horizontal and vertical velocities are shown in Fig. 7.7b. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare

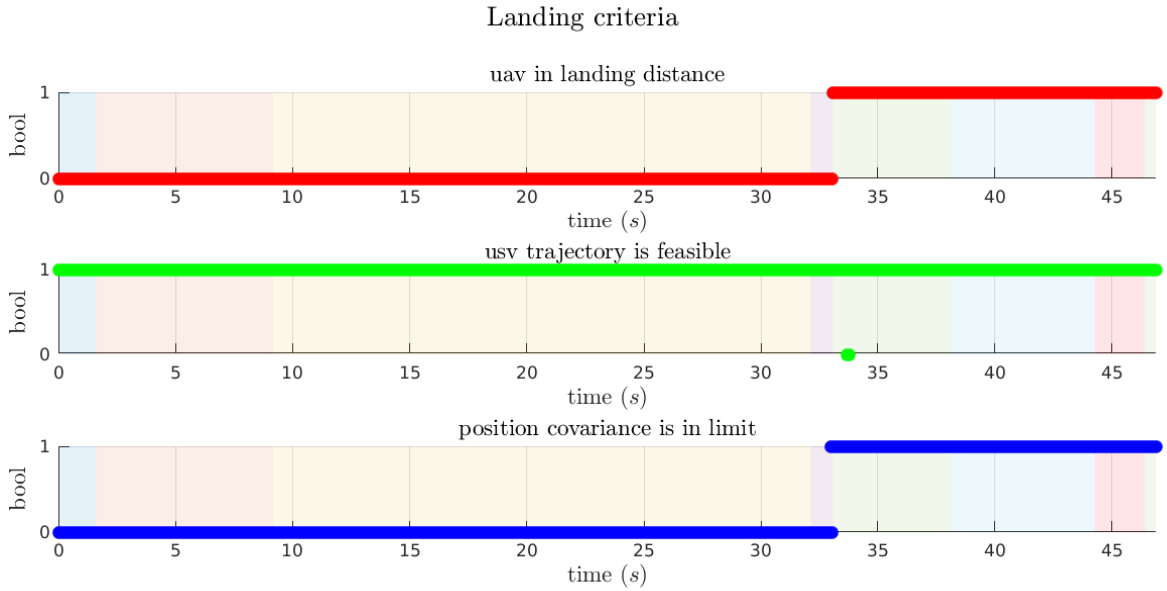


Figure 7.8: Landing criteria showing in which time which criterion was fulfilled. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

7.2.2 Evaluation

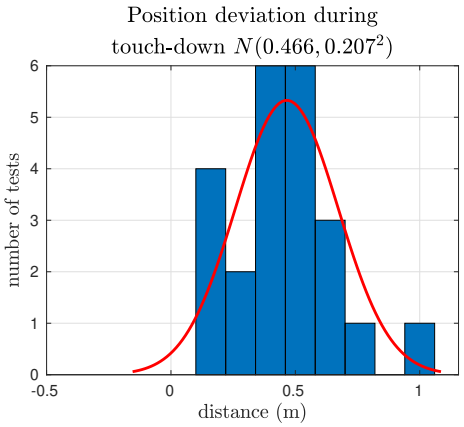
To verify the reliability of the proposed system the landing on the moving vessel, 23 experiments were performed. The vessel was assigned to follow the path's waypoints which created the shape of a large square and the shape of a figure-eight. Thanks to their shape, movement in various directions was ensured to examine the reliability of the USV's estimation and prediction. The simulation environment and trajectory generator were set up based on the Table 7.4.

Symbol	Value	Description
T_p	$5\,\mathrm{s}, 7\,\mathrm{s}$	wave's peak period
A	$0.2\,\mathrm{m}, 0.5\,\mathrm{m}$	wave's amplitude
v_{la}	$1\,\mathrm{m\,s^{-1}}$	landing approach velocity
v_{fa}	$0.5\,\mathrm{m\,s^{-1}}$	flare approach velocity
v_b	$2.0\,\mathrm{m\,s^{-1}}$	vessel's velocity

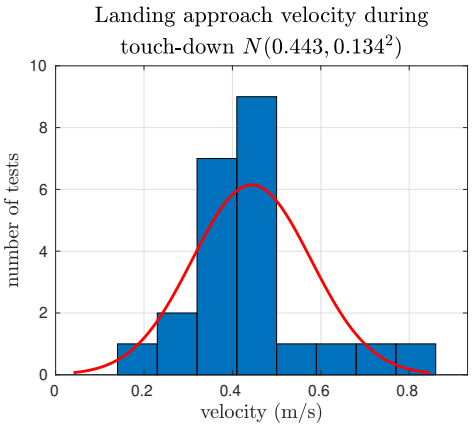
Table 7.4: Simulation settings for path-following vessel.

The resulting graphs are depicted in Fig. 7.9. According to the Fig. 7.9a, it can be seen that the mean value of the landing error was 0.466 m. This means that the UAV landed periodically on the edge of the area, which is covered with the AprilTag. In comparison with the results that were obtained in Sec. 7.1 the position error mean increased by about 31 cm. This can be caused by the inaccurate prediction or due to the UAV's acceleration limits to follow the trajectory.

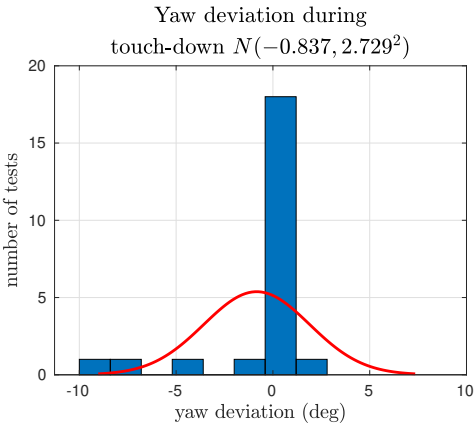
In Fig. 7.9b, the landing approach speed is shown. The mean value is $0.443\,\mathrm{m\,s^{-1}}$, which is $0.057\,\mathrm{m\,s^{-1}}$ smaller than v_{fa}. Nevertheless, this can be due to measurement error.



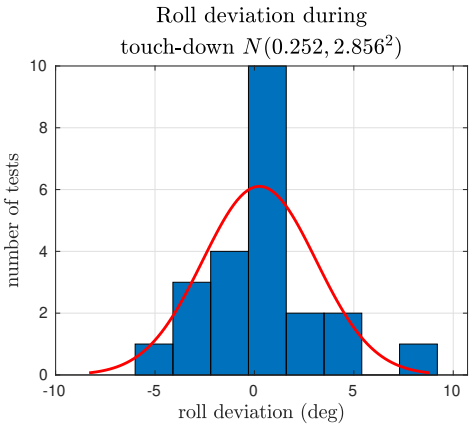
(a) UAV's position deviation from USV's centre during touch-down.



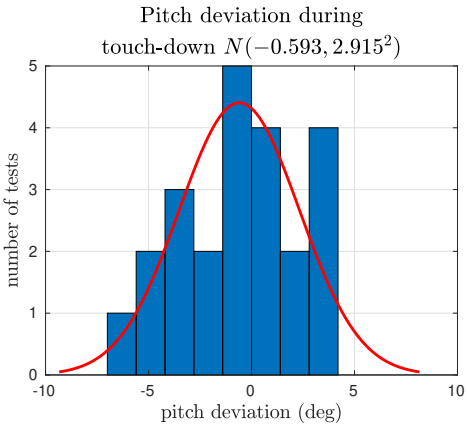
(b) UAV's approach speed during touch-down.



(c) UAV's yaw deviation from USV's during touch-down.



(d) UAV's roll deviation from USV's during touch-down.



(e) UAV's pitch deviation from USV's during touch-down.

Figure 7.9: Landing on a path-following vessel verification.

An interesting observation can be made in Fig. 7.9c which shows yaw deviation. The mean value is close to zero, and most tests ended with a yaw deviation between ±1 degree. However, values smaller than −5° were also appearing. This is probably caused by the fact that some landing attempts were performed during the USV's turning manoeuvre and the UAV was slightly lagging behind. Whereas roll angle deviation (see Fig. 7.9d) seems to follow the normal distribution with a mean value equal to 0.252° and standard deviation equal to 2.856°. The pitch angle deviation (see Fig. 7.9e) behaves more randomly. It is worth mentioning that it is expected that the pitch angle deviation will be larger due to the fact that the tilt around the y-axis is responsible for forward/backward velocity. However, for better evaluation, more tests are required.

Summary

It has been shown in the aforementioned experiments that the proposed trajectory generator is capable of generating trajectories for landing on a path-following boat. The sea, during the experiments, can be described as slight or moderate.

Based on these experiments it was observed that the accuracy of predicted states of the USV's depends on the direction of the vessel's movement. For example, if the vessel's direction was against the waves, the predictions of the USV's states were more accurate, and thus, the landing was more precise.

The trajectory generator also shows its limits. For example, the system, which should dynamically change linear velocities, was not tuned due to the absence of soft constraints. Moreover, the slight difference between the trajectory generated with the proposed system and the trajectory tracker was causing inaccuracy during the landing as well as in the experiments mentioned in Sec. 7.1.1.

Chapter 8

Real-world experiments

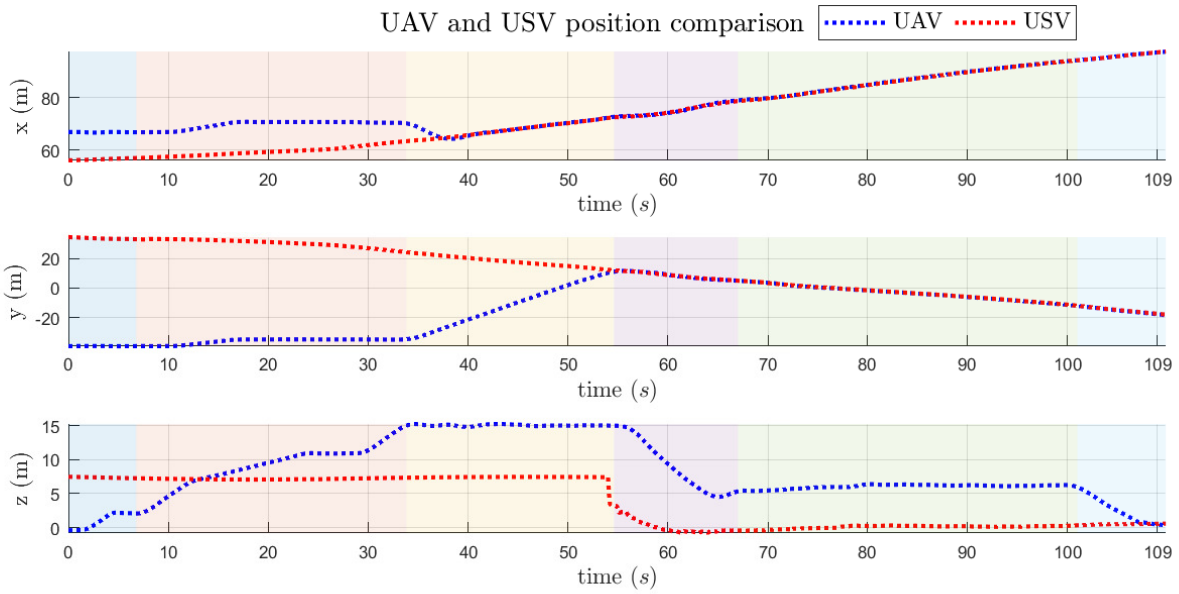
To verify the results that were obtained from simulations, two real-world experiments were performed. Both experiments took place on Orlík dam on the Vltava river (Fig. 8.1). Since the water is very calm there, and we wanted to test landing on a rocking boat, there was a need to rock the boat using human power. Naturally, this simplifies the vessel's movements on a water and can not be compared with real-world waves or even with the waves in a simulator environment. Due to the calm nature of the body of water, it was impossible to simulate the heaving motion in which the boat changes altitude compared to the water surface when still. It was impossible to induce such movements that the boat should heave to change altitude against the calm water surface while the vessel was rocking. Therefore, landing on rough sea was possible to perform only in simulations (see Sec. 7.1.1). The experiments included autonomous landing on a rocking vessel's deck, tracking the boat at a predefined height and even fully autonomous flight from the UAV's take-off to landing on the moving boat. The UAV and USV, which were described in Sec. 2.5.1, were used for real-world experiments.

The first round of experiments was performed together with testing the estimation and prediction of the USV's states. The results are shown in [24, p. 54-61]. This information was given to the trajectory planner responsible for creating the feasible trajectory for the UAV to approach, track and land on the vessel's deck. Since the first experiments demonstrated only simple solutions of the trajectory planner, we will refer to the results mentioned in [24]. The state machine used in the trajectory planner was significantly improved and the MPC based trajectory generator was tuned better thanks to these experiments. The prediction horizon was increased as well as the generating trajectory frequency.

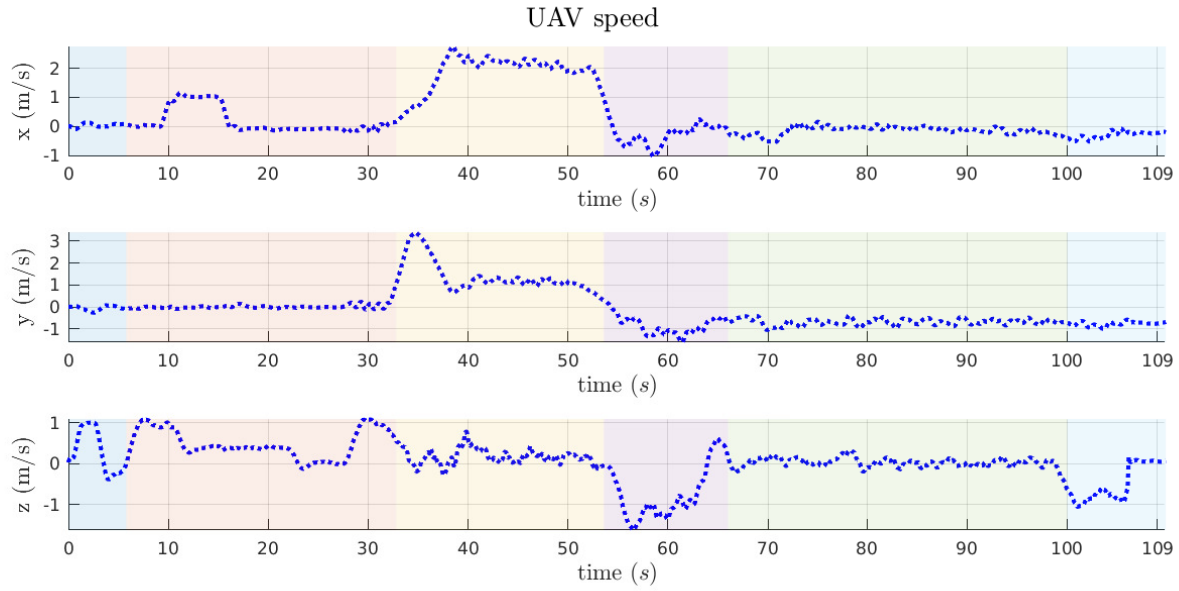
The second round of the experiments achieved better results. It verified that the trajectory planner proposed in this thesis in Chap. 5, together with the estimation and prediction of the vessel's states, mentioned in [24], is suitable for autonomous landing on the vessel in severe environment. The experiments and their results declaring this statement are presented in the following graphs and pictures.

Figure 8.1: Orlík dam on the Vltava river with a map showing the exact location where the tests were performed. (map source: Mapy.cz).

In Fig. 8.2, a fully autonomous test is presented, which is an experiment including all events that may occur during planning of the trajectory from the UAV's take-off to landing on the boat's deck. The weather conditions were measured in the nearest weather recording



(a) Comparison of the UAV's and USV's position.



(b) UAV's linear speed in body-fixed frame.

Figure 8.2: Comparison of the UAV's position and USV's estimated position is shown in Fig. 8.2a. UAV's speed is displayed in Fig. 8.2b. The Touch-down was performed at 107 seconds from the start. The colour marks individual states in which the UAV was during the test. The meaning of the background colours is following — dark blue: take-off, orange: get height, yellow: approach, purple: descent for tracking, green: tracking USV and light blue: landing. The graphs showing the UAV's and USV's attitudes are enclosed in Appx. C.

station TEMELIN, EZ. According to the weather history[1], the temperature was around 27 °C and wind was circa 7 km h^{-1} with wind gusts up to 17 km h^{-1}. Such wind speed is described as a Light breeze, according to the wind specification mentioned in [20, p. 13]. Since the measured data were recorded 20 km far away, the actual weather conditions could have been slightly different during the experiment.

The first phase is the idle stage, when the trajectory planner awaits the end of the take-off manoeuvre. After that, control was given to the safety pilot to verify that all the necessary systems works well. During the verification, the pilot tested the UAV's control and climbed to 11 m. From that moment on, control was given to the trajectory planner, which started generating the trajectory to climb to 15 m above the ground. The UAV started tracking the generated trajectory immediately. The approach manoeuvre started at time $t = 33$ s

(a) Get height manoeuvre at time $t = 25$ s. (b) Approach manoeuvre at time $t = 35$ s.

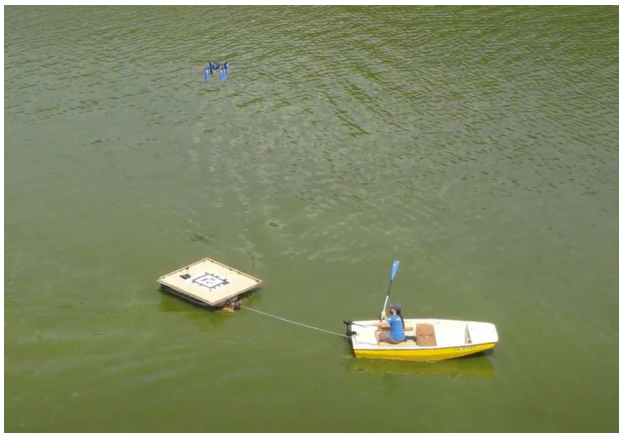
Figure 8.3: Snapshots from the real-world experiment — get height and approach phases

and continued for 21 s. The GPS position of the USV was provided by Nimbro network communication, which is a ROS transport for high-latency, low-quality networks. Therefore, the trajectory planner knew where to fly. However, the heading control was turned off for this experiment. The photo of the flying UAV to the USV's position can be seen in Fig. 8.3. From the Fig. 8.2b is evident that the trajectory tracker was limited to fly only 2 m s^{-1} in both x and y axes and the vertical speed was limited to ± 1 m s^{-1}.

According to the Fig. 8.2a, the UAV started descending to a predefined tracking height at 54 s. Moreover, the estimated z-position of the boat was modified due to the updates of the height received from the AprilTag detector and UVDAR system. The measurements from these two systems were given to the Kalman filter, thus the vessel's z-position was updated. The tracking height above the boat's deck was set to 5 meters. The USV was tracked from time $t = 67$ s to time $t = 101$ s when the landing phase started. The snapshots from the camera and DJI drone during the tracking and landing are shown in Fig. 8.4.

The landing stage started at time $t = 101$ s when all conditions for landing were met. The UAV was flying stable above the vessel's deck for some time, and the predicted states of the USV were within limits. The absolute value of roll and pitch USV's angles were smaller

[1] https://www.visualcrossing.com/weather-history/Jet%C4%9Btick%C3%A9%20samoty/metric/2022-07-22/2022-07-22



(a) Tracking the boat at time $t = 70\,\mathrm{s}$.

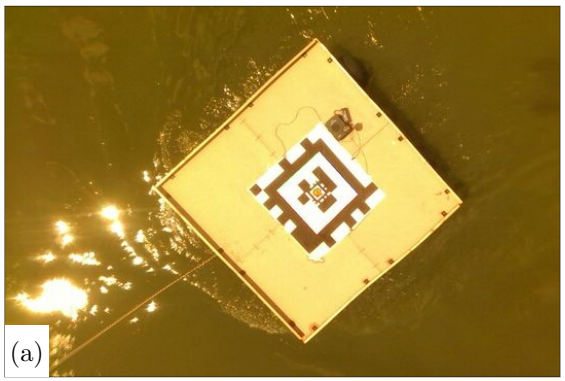
(b) Tracking the boat at time $t = 85\,\mathrm{s}$.

(c) Landing on the boat at time $t = 106\,\mathrm{s}$.

(d) Landing on the boat at time $t = 106\,\mathrm{s}$.

Figure 8.4: Snapshots from the real-world experiment — tracking and landing phases.

than 10°, and the USV's velocity did not exceed the limitations that were set up on UAV's trajectory tracker. The UAV started descending at a speed $1\,\mathrm{m\,s^{-1}}$ towards the vessel's landing platform. The views from the RealSense camera and UV camera during the landing are shown in Fig. 8.5. From these pictures can be seen that both AprilTags (the smaller one is placed in the middle of the bigger one) were detected and also the three of five UV LEDs were identified. The estimated roll angle varied between $\pm 1°$ and pitch angle varied between $3.7 - 5$ degrees



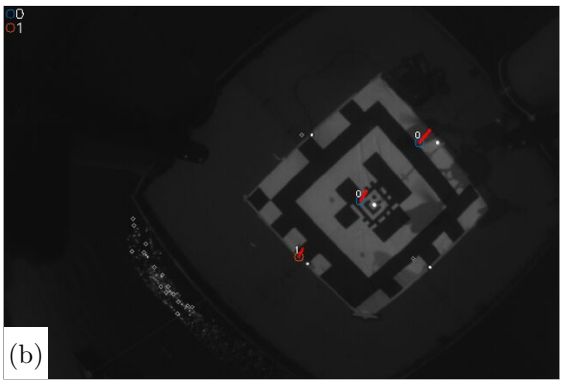


Figure 8.5: View from the UAV to the landing platform with (a) RealSense camera detecting the AprilTag and (b) UVDAR camera detecting the position of the UV LEDs during landing.

during the landing (see Appx. C.1b). Successful touch-down was identified at time $t = 107\,\text{s}$. The position of the UAV on the vessel's landing platform after the landing is shown in Fig. 8.6. The photo shows that the UAV landed almost into the centre of the deck. The deviation of the position from the centre of the AprilTag was around 15 cm.

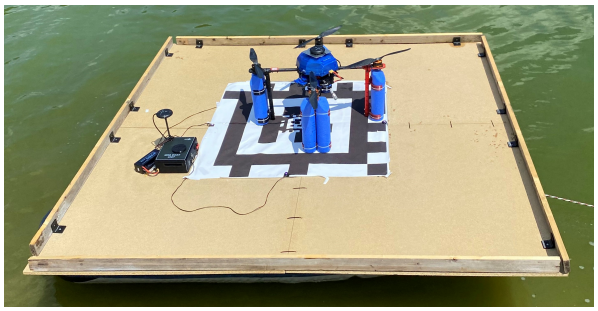


Figure 8.6: The position of the UAV after landing on the boat's deck during experiments on Orlík dam.

8.1 Evaluation

In the first and second rounds of the experiments, a few imperfections were revealed. One of the problems was the estimation of the USV's velocity, which was inaccurately predicted to be smaller than it was in reality (see Fig. C.2). Therefore, the predictions of the USV's future positions were wrongly predicted, and thus, the UAV was slightly lagging behind the vessel's deck centre. Other problems were induced oscillations and velocities, which cause an error in estimation. Nevertheless, it is also worth mentioning that none of the UAVs were drowned during both experiment rounds.

After the real-world experiments, we still saw an opportunity to improve the proposed solution. Therefore, the MPC has undergone further tuning, and other improvements to the trajectory planner have been made. These improvements were not tested in the real-world, but only in simulations shown in the previous chapter Chap. 7.

Chapter 9

Conclusion

This thesis aimed to develop a system for autonomous trajectory planning for landing an unmanned multirotor helicopter on top of a moving vessel. The developed trajectory generator was built on a system that provides accurate estimation and prediction of vessel's states, as well as a system of precise autonomous control of a UAV. For generating trajectories, the full-state tracker based on MPC was designed.

The MPC was implemented using OSQP solver, which belongs to a group of solvers that use first-order methods for solving the optimisation problem. These methods are usually computationally cheap and require a small number of iterations, even for large-scale optimisation problems. Thanks to that, a relatively accurate model of the UAV, consisting of twelve states describing its dynamics, could be used.

The strategies that were developed, during this project, ensure that the trajectory generator is capable of creating trajectories from take-off to touch-down. The mission and navigation state machine developed in this thesis is divided into three parts which are designed as approach, follow and landing. Each of these phases is responsible for the creation of the full-state reference, which is supplied to the full-state tracker. Special attention was given to the landing phase. The final landing manoeuvre was inspired by the flare technique well-known in the aerospace industry. Therefore, just before the UAV touches down, the approach speed is reduced, and attitude angles are aligned with the USV to decrease the impact of the UAV's legs. Thanks to that, the UAV is capable of landing on a moderate sea.

In the first phase, the developed system was tested in MATLAB and then incorporated into the MRS UAV control system. After that, the system was tested in various experiments in a realistic Gazebo simulation. Furthermore, two rounds of real-world experiments were conducted.

According to the thesis assignment, all these tasks have been successfully completed:

- The author of this thesis familiarised himself with the MRS UAV system for control, estimation, and simulation of multirotor helicopters is described in Chap. 2.
- Existing approaches of autonomous trajectory generation and planning for autonomous landing of a UAV on top of a moving USV were studied in Chap. 1.
- A realistic simulation environment in the Gazebo simulator that includes a catamaran boat was prepared and is described in Chap. 6. The boat's path-following feedback controller for automation during the development was designed in Chap. 6.
- The estimated states needed for autonomous landing on the boat were defined as position, attitude angles and linear velocities.
- The trajectory generator was designed based on the MPC full-state tracker mentioned in Chap. 3. The way of preparing the trajectory for the autonomous landing of the UAV on the top of a moving catamaran was designed in Chap. 5.
- The proposed system using the simulation environment was verified and the results were shown in Chap. 7. Moreover, the simplified system was demonstrated by conducting real-

world experiments using a real-world robotic setup which was mentioned in Chap. 1. Results of these experiments were presented in Chap. 8.

9.1 Future work

A few imperfections in the proposed system were revealed during the development of this work. Firstly, the designed mathematical model of the UAV does not model accelerations. However, this does not apply for the MRS trajectory tracker, and therefore, the resulting trajectory is not followed with expected precision. This deficiency should be resolved by including the accelerations into the UAV model. Unfortunately, the computation complexity will increase due to an increased amount of states. To solve this problem, the implementation of the OSQP solver on a Graphics Processing Unit (GPU) can be used [114]. Moreover, if we wanted to use the proposed system without the MRS trajectory tracker, soft constraints would have to be implemented. Last but not least, it would be beneficial to tune and test the proposed system in a more challenging environment.

Chapter 10

References

[1] J. Scherer, S. Yahyanejad, S. Hayat, *et al.*, "An Autonomous Multi-UAV System for Search and Rescue," in *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, ser. DroNet '15, Florence, Italy: Association for Computing Machinery, 2015, 33–38.

[2] P. Doherty and P. Rudol, "A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization," in *AI 2007: Advances in Artificial Intelligence*, M. A. Orgun and J. Thornton, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 1–13.

[3] N. Sharma, M. Saqib, P. Scully-Power, and M. Blumenstein, "SharkSpotter: Shark Detection with Drones for Human Safety and Environmental Protection," in *Humanity Driven AI: Productivity, Well-being, Sustainability and Partnership*, F. Chen and J. Zhou, Eds. Cham: Springer International Publishing, 2022, pp. 223–237.

[4] A. Akib, F. Tasnim, D. Biswas, *et al.*, "Unmanned Floating Waste Collecting Robot," in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, 2019, pp. 2645–2650.

[5] *Speciální záchranářské Drony Budou Varovat Australské Plavce před žraloky*, 2021. [Online]. Available: https://www.stoplusjednicka.cz/specialni-zachranarske-drony-budou-varovat-australske-plavce-pred-zraloky.

[6] 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.

[7] X. Huang, Q. Xu, and J. Wang, "Vision-based Autonomous Landing of UAV on Moving Platform using a New Marker," *IOP Conference Series: Materials Science and Engineering*, vol. 646, no. 1, p. 012 062, 2019.

[8] E. Olson, "AprilTag: A robust and flexible visual fiducial system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2011, pp. 3400–3407.

[9] H. Jiaxin, Y. Guo, F. Zhen, and G. Yuqing, "Vision-based autonomous landing of unmanned aerial vehicles," Oct. 2017, pp. 3464–3469.

[10] Z.-C. Xu, B.-B. Hu, B. Liu, X. Wang, and H.-T. Zhang, "Vision-based Autonomous Landing of Unmanned Aerial Vehicle on a Motional Unmanned Surface Vessel," in *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 6845–6850.

[11] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[12] A. Keller and B. Ben-Moshe, "A Robust and Accurate Landing Methodology for Drones on Moving Targets," *Drones*, vol. 6, no. 4, 2022.

[13] L. Yang, Z. Liu, X. Wang, G. Wang, X. Hu, and Y. Xi, "Autonomous Landing of a Rotor Unmanned Aerial Vehicle on a Boat Using Image-Based Visual Servoing," in *2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2021, pp. 1848–1854.

[14] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[15] J. Rawlings, "Tutorial overview of model predictive control," *IEEE Control Systems Magazine*, vol. 20, no. 3, pp. 38–52, 2000. DOI: 10.1109/37.845037.

[16] K. Guo, P. Tang, H. Wang, D. Lin, and X. Cui, "Autonomous Landing of a Quadrotor on a Moving Platform via Model Predictive Control," *Aerospace*, vol. 9, no. 1, 2022.

[17] L. Persson and B. Wahlberg, "Model Predictive Control for Autonomous Ship Landing in a Search and Rescue Scenario," in *AIAA Scitech 2019 Forum*.

[18] T. D. Ngo and C. Sultan, "Nonlinear Helicopter and Ship Models for Predictive Control of Ship Landing Operations," in *AIAA Guidance, Navigation, and Control Conference*.

[19] S. Abujoub, J. McPhee, and R. A. Irani, "Methodologies for landing autonomous aerial vehicles on maritime vessels," *Aerospace Science and Technology*, vol. 106, p. 106 169, 2020.

[20] S. Abujoub, "Development of a landing period indicator and the use of signal prediction to improve landing methodologies of autonomous unmanned aerial vehicles on maritime vessels," Ph.D. dissertation, Carleton University, 2019.

[21] S. Abujoub, J. McPhee, C. Westin, and R. A. Irani, "Unmanned Aerial Vehicle Landing on Maritime Vessels using Signal Prediction of the Ship Motion," in *OCEANS 2018 MTS/IEEE Charleston*, 2018, pp. 1–9.

[22] A. Paris, B. T. Lopez, and J. P. How, *Dynamic Landing of an Autonomous Quadrotor on a Moving Platform in Turbulent Wind Conditions*, 2019.

[23] N. Muraleedharan and D. S. Cohen, "Modelling and simulation of UAV systems," *Imaging and Sensing for Unmanned Aircraft Systems: Control and Performance, Volume 1*, p. 101, 2020.

[24] F. Novak, "State Estimation of an Unmanned Surface Vehicle by an Unmanned Multirotor Helicopter," M.S. thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2022.

[25] K. Williams, T. Honeyands, R. Holmes, *et al.*, "Maritime Bulk Cargo Transportable Moisture Limit Requirements for Iron Ore Shipments," Jul. 2015.

[26] T. I. Fossen and T. Perez, "Kalman filtering for positioning and heading control of ships and offshore rigs," *IEEE Control Systems Magazine*, vol. 29, no. 6, pp. 32–46, 2009.

[27] T. I. Fossen, in *Handbook of Marine Craft Hydrodynamics and Motion Control*. 2011.

[28] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[29] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, p. 46, 2004.

[30] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*, Spie, vol. 3068, 1997, pp. 182–193.

[31] V Silva and S Parkes, "On the Design of an Optimal GNC Sensor Architecture for Autonomous Planetary Landers," in *DASIA 2003-Data Systems In Aerospace*, vol. 532, 2003.

[32] V. Walter, N. Staub, A. Franchi, and M. Saska, "UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2637–2644, 2019.

[33] V. Walter, M. Vrba, and M. Saska, "On training datasets for machine learning-based visual relative localization of micro-scale UAVs," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[34] V. Walter, N.Staub, M. Saska, and A. Franchi, "Mutual Localization of UAVs based on Blinking Ultraviolet Markers and 3D Time-Position Hough Transform," in *14th IEEE International Conference on Automation Science and Engineering (CASE 2018)*, 2018.

[35] V. Walter, M. Saska, and A. Franchi, "Fast mutual relative localization of uavs using ultraviolet led markers," in *2018 International Conference on Unmanned Aircraft System (ICUAS 2018)*, 2018.

[36] T. Baca, P. Stepan, and M. Saska, "Autonomous landing on a moving car with unmanned aerial vehicle," in *2017 European Conference on Mobile Robots (ECMR)*, 2017, pp. 1–6.

[37] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model Predictive Trajectory Tracking and Collision Avoidance for Reliable Outdoor Deployment of Unmanned Aerial Vehicles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 6753–6760.

[38] T. Roucek, M. Pecka, P. Cizek, *et al.*, "System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge," *Field Robotics*, vol. 2, pp. 1779–1818, 2022.

[39] V. Walter, V. Spurny, M. Petrlik, T. Baca, D. Zaitlik, and M. Saska, "Extinguishing of Ground Fires by Fully Autonomous UAVs Motivated by the MBZIRC 2020 Competition," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2021, pp. 787–793.

[40] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control: Technical report," ETH Zurich, Tech. Rep., 2013.

[41] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[42] D. Hert, "Autonomous Predictive Interception of a Flying Target by an Unmanned Aerial Vehicle," M.S. thesis, Prague, Czech Republic: Czech Technical University in Prague, 2018.

[43] T. Schwickart, H. Voos, M. Darouach, and S. Bezzaoucha, "A flexible move blocking strategy to speed up model-predictive control while retaining a high tracking performance," in *2016 European Control Conference (ECC)*, IEEE, 2016, pp. 764–769.

[44] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*, Springer, 2016, pp. 649–666.

[45] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart, "Real-Time Visual-Inertial Mapping, Re-localization and Planning Onboard MAVs in Unknown Environments," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2015.

[46] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[47] I. Cermak, "Unmanned Aerial Vehicle Design and Sensor Integration for Flying over Water Area," Bachelor's thesis, České vysoké učení technické v Praze. Vypočetní a informační centrum., 2022.

[48] D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.

[49] S. J. Qin and T. A. Badgwell, "An overview of industrial model predictive control technology," in *AIche symposium series*, New York, NY: American Institute of Chemical Engineers, 1971-c2002., vol. 93, 1997, pp. 232–256.

[50] ——, "An overview of nonlinear model predictive control applications," *Nonlinear model predictive control*, pp. 369–392, 2000.

[51] K. Worthmann, "Estimates of the Prediction Horizon Length in MPC: a Numerical Case Study," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 232–237, 2012, 4th IFAC Conference on Nonlinear Model Predictive Control.

[52] M. N. Zeilinger, M. Morari, and C. N. Jones, "Soft constrained model predictive control with robust stability guarantees," *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1190–1202, 2014.

[53] D Limon, I Alvarado, T Alamo, and E. Camacho, "MPC for tracking of piece-wise constant references for constrained linear systems," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 135–140, 2005.

[54] A. Ferramosca, D. Limón, I. Alvarado, T. Alamo, and E. F. Camacho, "MPC for tracking with optimal closed-loop performance," in *2008 47th IEEE Conference on Decision and Control*, IEEE, 2008, pp. 4055–4060.

[55] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[56] M Islam, M Okasha, and M. Idres, "Dynamics and control of quadcopter using linear model predictive control approach," in *IOP Conference Series: Materials Science and Engineering*, IOP Publishing, vol. 270, 2017, p. 012 007.

[57] T. Baca, "Model predictive control of micro aerial vehicle using onboard microcontroller," M.S. thesis, Czech technical university in Prague, 2015.

[58] M. Zeilinger, C. Jones, and M. Morari, "Robust stability properties of soft constrained MPC," Jan. 2011, pp. 5276 –5282.

[59] R. Findeisen and F. Allgöwer, "An introduction to nonlinear model predictive control," in *21st Benelux meeting on systems and control*, Citeseer, vol. 11, 2002, pp. 119–141.

[60] G. Torrente, E. Kaufmann, P. Föhn, and D. Scaramuzza, "Data-driven MPC for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3769–3776, 2021.

[61] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart, "Fast nonlinear model predictive control for multicopter attitude tracking on so (3)," in *2015 IEEE Conference on Control Applications (CCA)*, IEEE, 2015, pp. 1160–1166.

[62] M. S. Kamel, M. Burri, and R. Siegwart, "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles," vol. 50, Jul. 2017.

[63] Z. Tahir and S. Liaqat, "State Space System Modelling of a Quad Copter UAV," Aug. 2019.

[64] D. Kotarski, P. Piljek, and K. Matija, "Mathematical Modelling of Multirotor UAV," 2016.

[65] A. Chovancová, T. Fico, Ľuboš Chovanec, and P. Hubinsk, "Mathematical Modelling and Parameter Identification of Quadrotor (a survey)," *Procedia Engineering*, vol. 96, pp. 172–181, 2014, Modelling of Mechanical and Mechatronic Systems.

[66] T. Luukkonen, "Modelling and control of quadcopter," *Independent research project in applied mathematics, Espoo*, vol. 22, p. 22, 2011.

[67] A. Sanca, P. Alsina, and J. Cerqueira, "Dynamic Modelling of a Quadrotor Aerial Vehicle with Nonlinear Inputs," *Latin American Robotics Symposium and Intelligent Robotics Meeting*, vol. 0, pp. 143–148, Oct. 2008.

[68] A. E. C. D. Cunha, "Benchmark: Quadrotor Attitude Control," in *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, G. Frehse and M. Althoff, Eds., ser. EPiC Series in Computing, vol. 34, EasyChair, 2015, pp. 57–72.

[69] B. Erginer and E. Altug, "Modeling and PD Control of a Quadrotor VTOL Vehicle," in *2007 IEEE Intelligent Vehicles Symposium*, 2007, pp. 894–899.

[70] P. Foehn, A. Romero, and D. Scaramuzza, "Time-optimal planning for quadrotor waypoint flight," *Science Robotics*, vol. 6, no. 56, eabh1221, 2021.

[71] G. V. Raffo, M. G. Ortega, and F. R. Rubio, "An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter," *Automatica*, vol. 46, no. 1, pp. 29–39, 2010.

[72] M. Jalón, "Non-linear Attitude Control and Guidance of a Quadrotor UAV," 2014.

[73] H. Huang, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Aerodynamics and control of autonomous quadrotor helicopters in aggressive maneuvering," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3277–3282.

[74] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[75] M. Bangura, R. Mahony, *et al.*, "Nonlinear dynamic modeling for high performance control of a quadrotor," 2012.

[76] L. Martins, C. Cardeira, and P. Oliveira, "Linear Quadratic Regulator for Trajectory Tracking of a Quadrotor," *IFAC-PapersOnLine*, vol. 52, no. 12, pp. 176–181, 2019, 21st IFAC Symposium on Automatic Control in Aerospace ACA 2019.

[77] M. De Lellis Costa de Oliveira, "Modeling, Identification and Control of a Quadrotor Aircraft," M.S. thesis, Jun. 2011.

[78] M. Jardin and E. Mueller, "Optimized Measurements of UAV Mass Moment of Inertia with a Bifilar Pendulum," *Journal of Aircraft - J AIRCRAFT*, vol. 46, pp. 763–775, May 2009.

[79] ——, "Optimized Measurements of UAV Mass Moment of Inertia with a Bifilar Pendulum," in *AIAA Guidance, Navigation and Control Conference and Exhibit*.

[80] M. Krznar, D. Kotarski, P. Piljek, and D. Pavković, "On-line Inertia Measurement of Unmanned Aerial Vehicles using on board Sensors and Bifilar Pendulum," *Interdisciplinary Description of Complex Systems*, vol. 16, pp. 149–161, Jan. 2018.

[81] A. E. C. da Cunha, "Benchmark: Quadrotor Attitude Control.," in *ARCH@ CPSWeek*, 2015, pp. 57–72.

[82] M. Quigley, "ROS: an open-source Robot Operating System," in *ICRA 2009*, 2009.

[83] W. Qian, Z. Xia, J. Xiong, *et al.*, "Manipulation task simulation using ROS and Gazebo," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, IEEE, 2014, pp. 2594–2598.

[84] C. E. Agüero, N. Koenig, I. Chen, *et al.*, "Inside the virtual robotics challenge: Simulating real-time robotic disaster response," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 494–506, 2015.

[85] M. Zhang, H. Qin, M. Lan, *et al.*, "A high fidelity simulator for a quadrotor UAV using ROS and Gazebo," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, IEEE, 2015, pp. 002 846–002 851.

[86] M. Krizek, J. Horyna, and M. Saska, "Swarming of Unmanned Aerial Vehicles by Sharing Distributed Observations of Workspace," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2022, pp. 300–309.

[87] B. Bingham, C. Aguero, M. McCarrin, *et al.*, "Toward Maritime Robotic Simulation in Gazebo," in *Proceedings of MTS/IEEE OCEANS Conference*, Seattle, WA, Oct. 2019.

[88] J. Tessendorf *et al.*, "Simulating ocean water," *Simulating nature: realistic and interactive techniques. SIGGRAPH*, vol. 1, no. 2, p. 5, 2001.

[89] J. Fréchot, "Realistic simulation of ocean surface using wave spectra," in *Proceedings of the first international conference on computer graphics theory and applications (GRAPP 2006)*, 2006, pp. 76–83.

[90] H. Mitsuyasu, F. Tasai, T. Suhara, *et al.*, "Observations of the directional spectrum of ocean WavesUsing a cloverleaf buoy," *Journal of Physical Oceanography*, vol. 5, no. 4, pp. 750–760, 1975.

[91] J. Pimentel, B. Molina, V. Sevillano, *et al.*, "Design of an Autonomous Surface Vehicle (ASV) for the 2022 Maritime Robotx Challenge,"

[92] Åström, Karl Johan and Hägglund, Tore, "The future of pid control," *Control engineering practice*, vol. 9, no. 11, pp. 1163–1175, 2001.

[93] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[94] G. Dantzig, "Linear Programming and Extensions, Princeton UniversityPress, 1963," *DantzigLinear Programming and Extensions1963*, 1963.

[95] M. ApS, "Mosek optimization toolbox for matlab," *User's Guide and Reference Manual, Version*, vol. 4, 2019.

[96] G. Optimization, "Inc., 2016. Gurobi optimizer reference manual," [Online]. Available: `http://www.gurobi.com`.

[97] H.J. Ferreau and H.G. Bock and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[98] H. J. Ferreau, "Model predictive control algorithms for applications with millisecond timescales," *Arenberg Doctoral School of Science, Engineering & Technology, Faculty of Engineering, Department of Electrical Engineering*, 2011.

[99] J. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, Dec. 2014.

[100] H. Ferreau, A. Potschka, and C. Kirches, *qpOASES webpage*, http://www.qpOASES.org/, 2007–2017.

[101] B. Houska, H. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[102] R. Verschueren, G. Frison, D. Kouzoupis, *et al.*, "Towards a modular software package for embedded optimization," in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018.

[103] R. Verschueren, G. Frison, D. Kouzoupis, *et al.*, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, 2021.

[104] A. Altman and J. Gondzio, "Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 275–302, 1999.

[105] R. J. Vanderbei and T. J. Carpenter, "Symmetric indefinite systems for interior point methods," *Mathematical programming*, vol. 58, no. 1, pp. 1–32, 1993.

[106] E. M. Gertz and S. J. Wright, "Object-oriented software for quadratic programming," *ACM Transactions on Mathematical Software (TOMS)*, vol. 29, no. 1, pp. 58–81, 2003.

[107] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[108] L. Blackmore, "Autonomous precision landing of space rockets," in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2016 Symposium*, The Bridge Washington, DC, vol. 46, 2016, pp. 15–20.

[109] P.-L. Lions and B. Mercier, "Splitting algorithms for the sum of two nonlinear operators," *SIAM Journal on Numerical Analysis*, vol. 16, no. 6, pp. 964–979, 1979.

[110] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[111] G. Banjac, P. Goulart, B. Stellato, and S. Boyd, "Infeasibility detection in the alternating direction method of multipliers for convex optimization," *Journal of Optimization Theory and Applications*, vol. 183, no. 2, pp. 490–519, 2019.

[112] J. H. Wilkinson, *Rounding errors in algebraic processes*. Courier Corporation, 1994.

[113] G. B. Bartolomeo Stellato, *OSQP Documentation, Release 0.3.1*, 2018. [Online]. Available: `https://buildmedia.readthedocs.org/media/pdf/osqp/stable/osqp.pdf`.

[114] M. Schubiger, G. Banjac, and J. Lygeros, "GPU acceleration of ADMM for large-scale quadratic programming," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 55–67, 2020.

Appendices

Appendix A

Tests of solvers for MPC control

To develop, tune and verify the full-states tracker based on MPC, the MATLAB standalone simulations were designed. Therefore, this appendix presents results from the MATLAB simulation experiments. Firstly, the UAV's nonlinear model dynamics were defined, and then this model was linearised and discretised with a sampling period $T_s = 0.1\,\mathrm{s}$. The full-state reference generator was developed for designing the trajectory using sine and cosine functions. The full-state reference was later replaced by measured data of the USV's states from the Gazebo simulator.

The experiments were designed to verify the MPC setting and also compare a qudprog solver, qpOASES solver and OSQP solver with respect to the length of the prediction horizon and the time duration of the one iteration step. The illustration of the UAV's trajectory together with the reference trajectory is shown in Fig. A.1. The starting point of the UAV was located far from the reference trajectory, but at the end, the UAV followed the reference trajectory successfully. Therefore, the test of solvers was divided into two sub-tasks, where the first one was to follow the trajectory from far but only for 5 s and the second one was to follow the reference trajectory for 20 s when the starting point was already on the trajectory.

The settings of the MPC are stated in Table A.1. Moreover, the qpOASES's default setting was limited only to 1000 iterations. The OSQP's default setting was limited to 4000 iterations, OSQP polishing was turned on, the absolute and relative tolerances were set to 10^{-5} and the check termination interval was set to 10.

Symbol	Value	Description
$\mathbf{Q}$	$[30, 30, 40, 1, 1, 30, 1, 1, 1, 1, 1, 50]$	diagonal objective function matrix
$\mathbf{P}$	$[30, 30, 40, 1, 1, 30, 1, 1, 1, 1, 1, 50]$	diagonal terminal penalty weight matrix
$\mathbf{R}$	$[0.1, 0.1, 0.1, 0.1]$	diagonal objective function matrix
$\phi,\ \theta$	$\pm 0.7854\,\mathrm{rad}$	roll and pitch min/max angle
$v_x,\ v_y$	$\pm 5\,\mathrm{m\,s^{-1}}$	horizontal min/max velocity in x and y axis
v_z	$\pm 5\,\mathrm{m\,s^{-1}}$	vertical min/max velocity
$v_\phi,\ v_\theta$	$\pm 5\,\mathrm{rad\,s^{-1}}$	roll and pitch rotation min/max velocity
v_ψ	$\pm 5\,\mathrm{rad\,s^{-1}}$	yaw rotation min/max velocity

Table A.1: Model Predictive Control settings for MATLAB simulations.

In order to be able to compare the obtained results in the future, the hardware specification of the computer on which the experiments were performed must be noted. The processor unit was i7-8550U, and the installed Physical Memory was 16.0 GB. The resulting graphs are shown in Fig. A.2, Fig. A.3 and Fig. A.4.

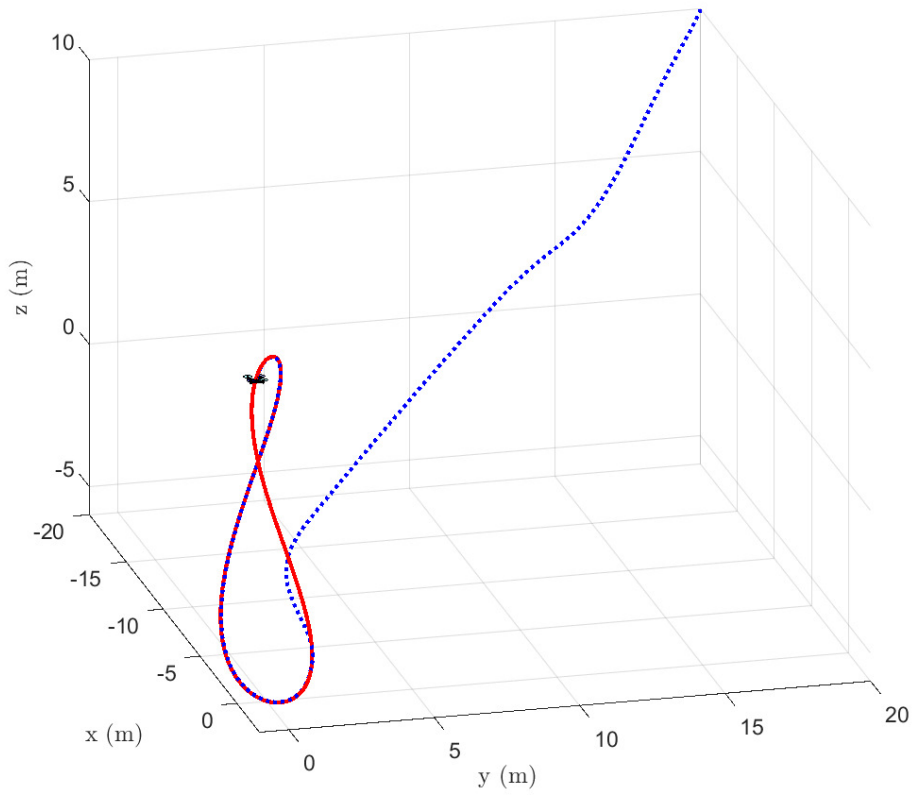
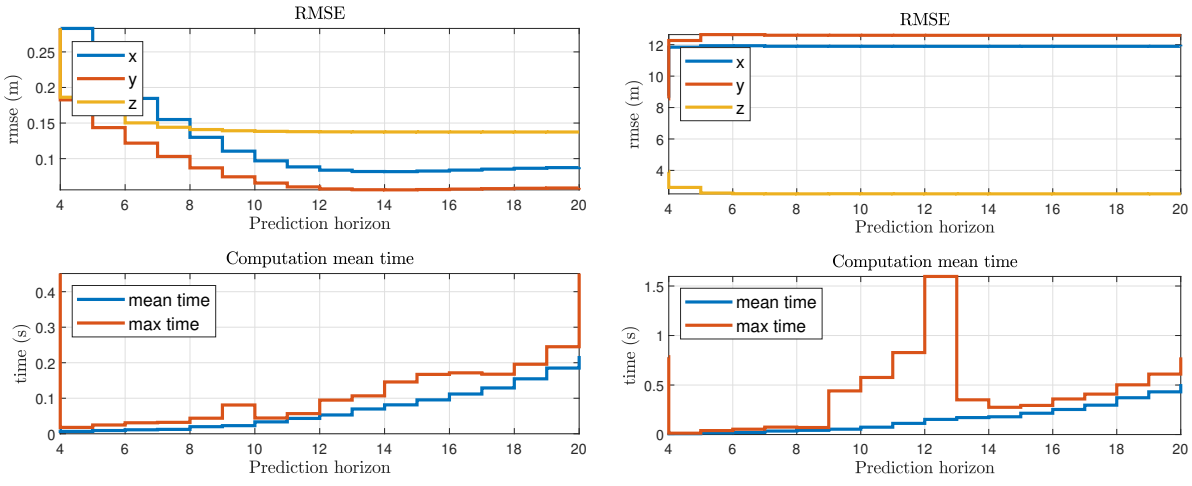


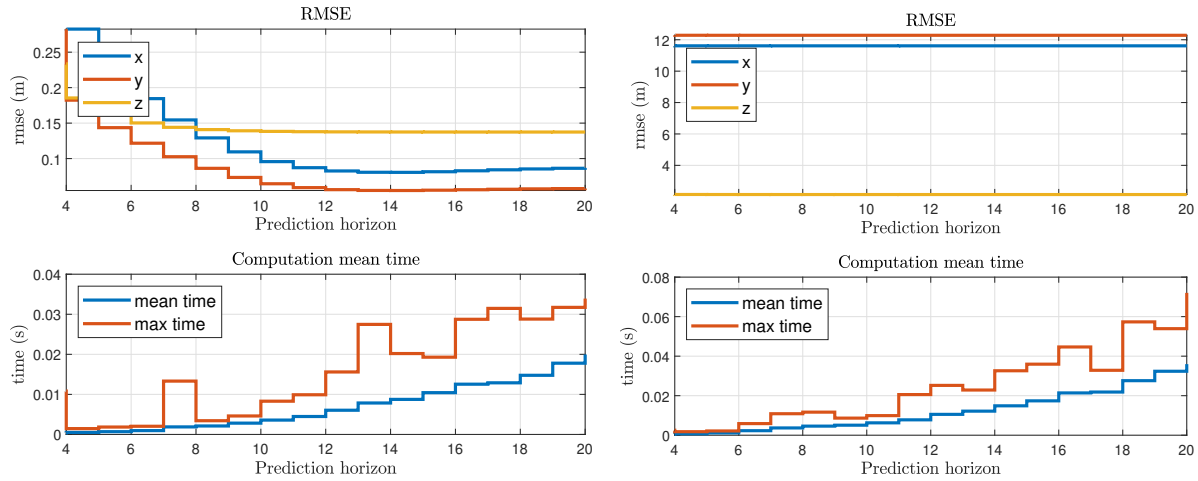
Figure A.1: The graph from the MATLAB simulation. The UAV's trajectory is marked with a blue dotted line, and the path which was followed is marked with a red colour line.



(a) Tracking trajectory test.

(b) Tracking trajectory, when the trajectory is far.

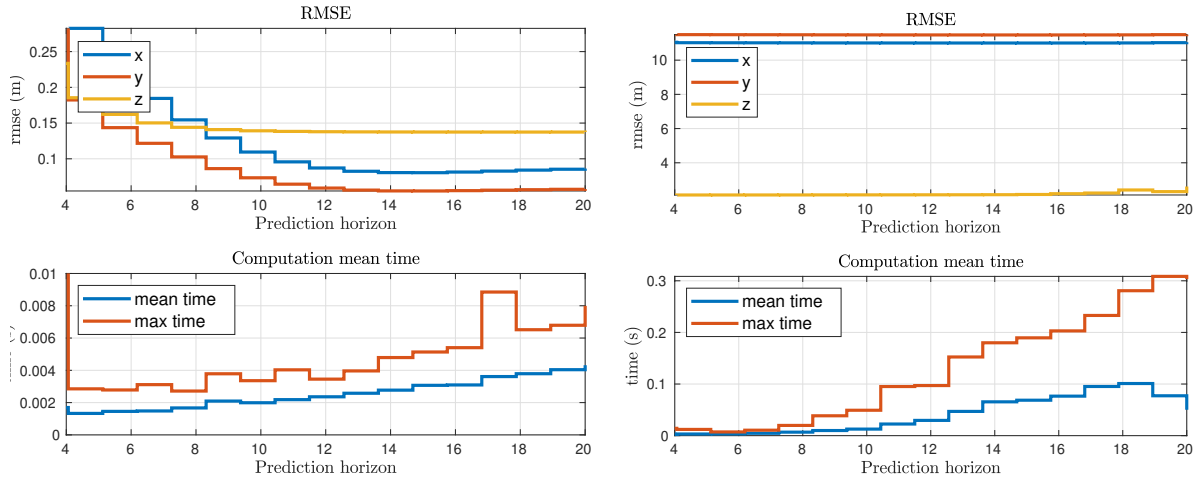
Figure A.2: Comparison of time duration and RMSE versus prediction horizon using quadprog.



(a) Tracking trajectory test.

(b) Tracking trajectory, when the trajectory is far.

Figure A.3: Comparison of time duration and RMSE versus prediction horizon using qpOASES.



(a) Tracking trajectory test.

(b) Tracking trajectory, when the trajectory is far.

Figure A.4: Comparison of time duration and RMSE versus prediction horizon using OSQP.

Appendix B

Simulations verification

B.1 Landing on a vessel on a moderate sea

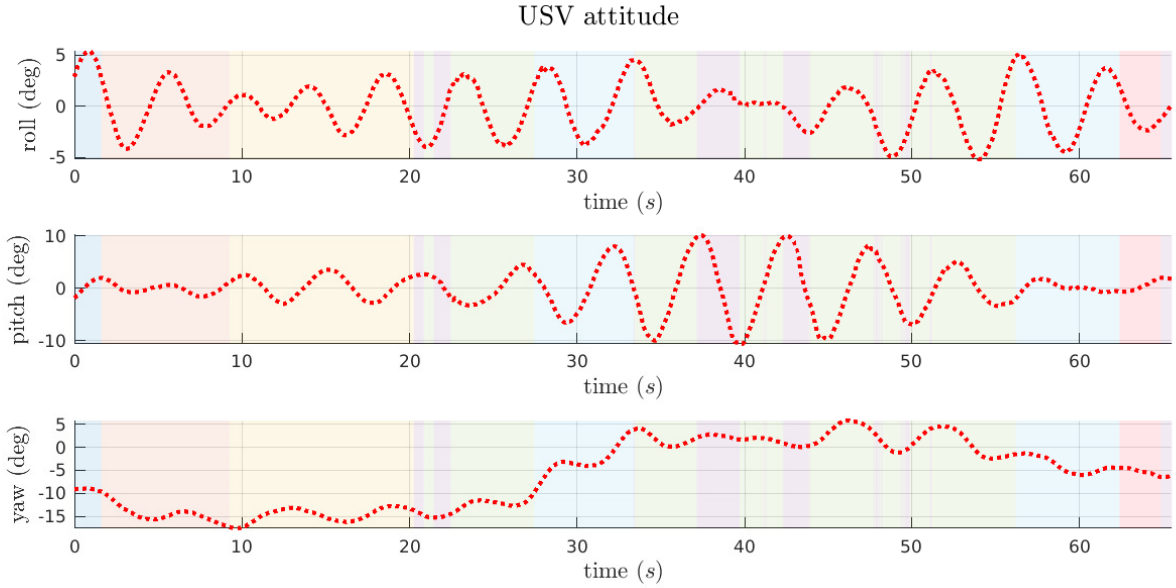


Figure B.1: USV's attitude angles displays roll, which was varying between ±5°, pitch angle was varying between ±10° and yaw was varying between −17.5° and 5.6°. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

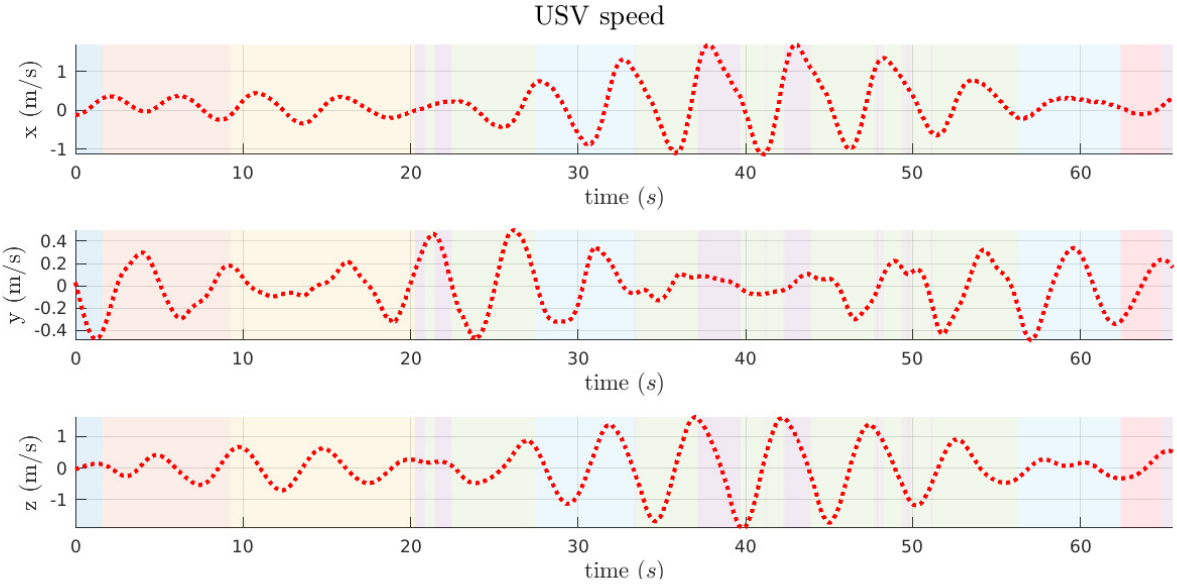
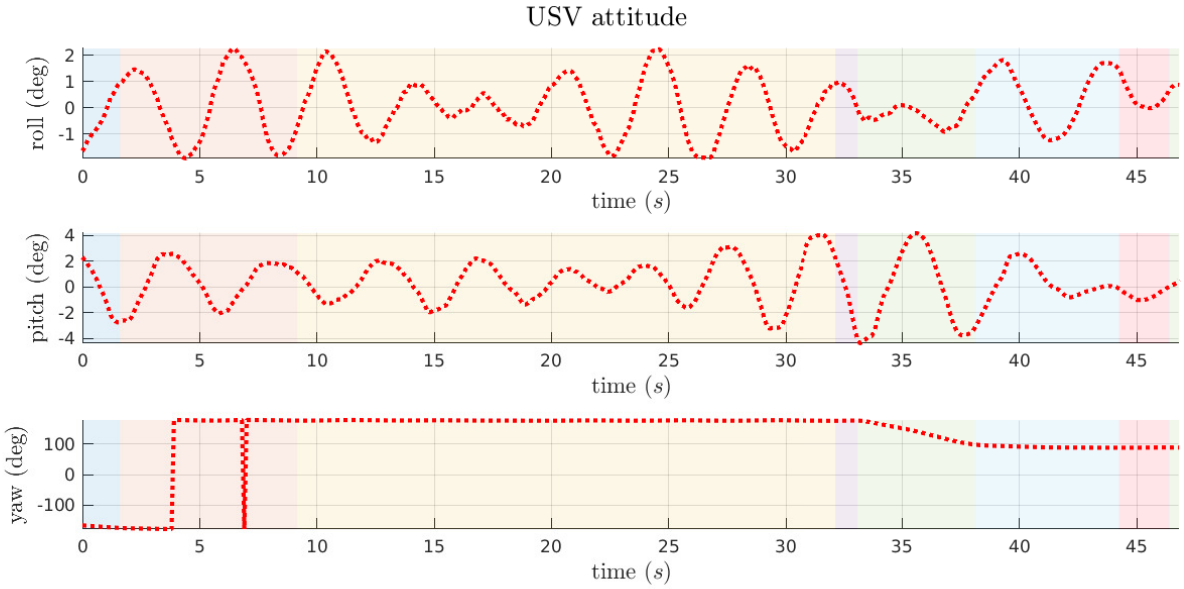
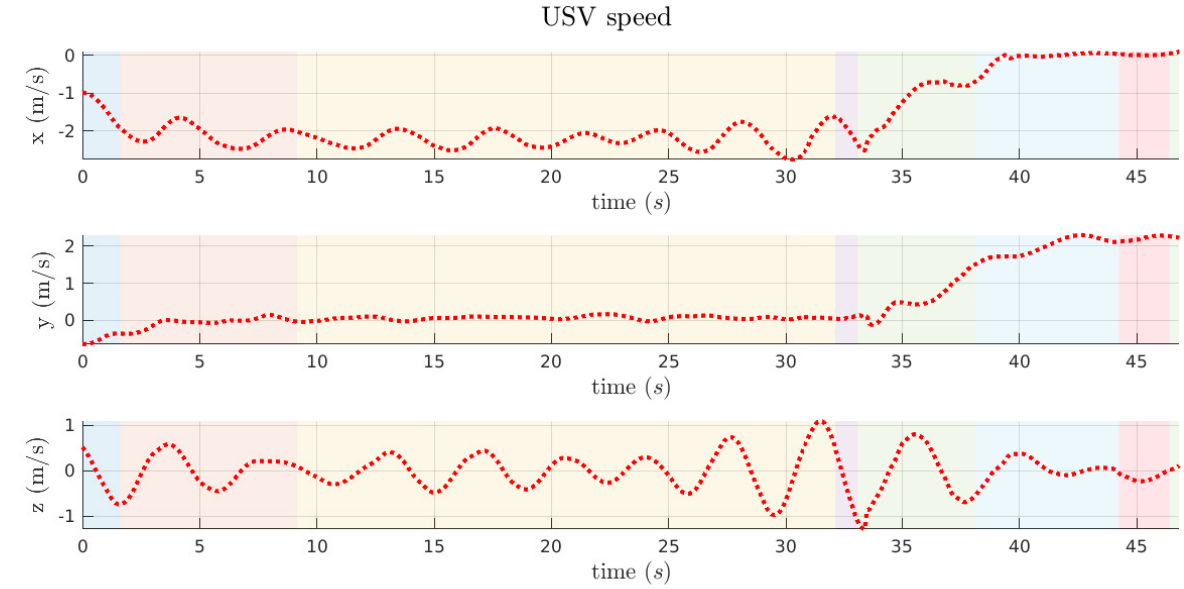


Figure B.2: USV's linear velocities in the global coordinate frame. The linear velocity in x-axis was varying between $-1.1\,\mathrm{m\,s^{-1}}$ and $1.6\,\mathrm{m\,s^{-1}}$, velocity in y-axis was varying between $-0.45\,\mathrm{m\,s^{-1}}$ and $0.48\,\mathrm{m\,s^{-1}}$ and vertical velocity was between $-1.9\,\mathrm{m\,s^{-1}}$ and $1.6\,\mathrm{m\,s^{-1}}$. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

B.2 Landing on a path-following vessel



(a) USV's attitude angles.

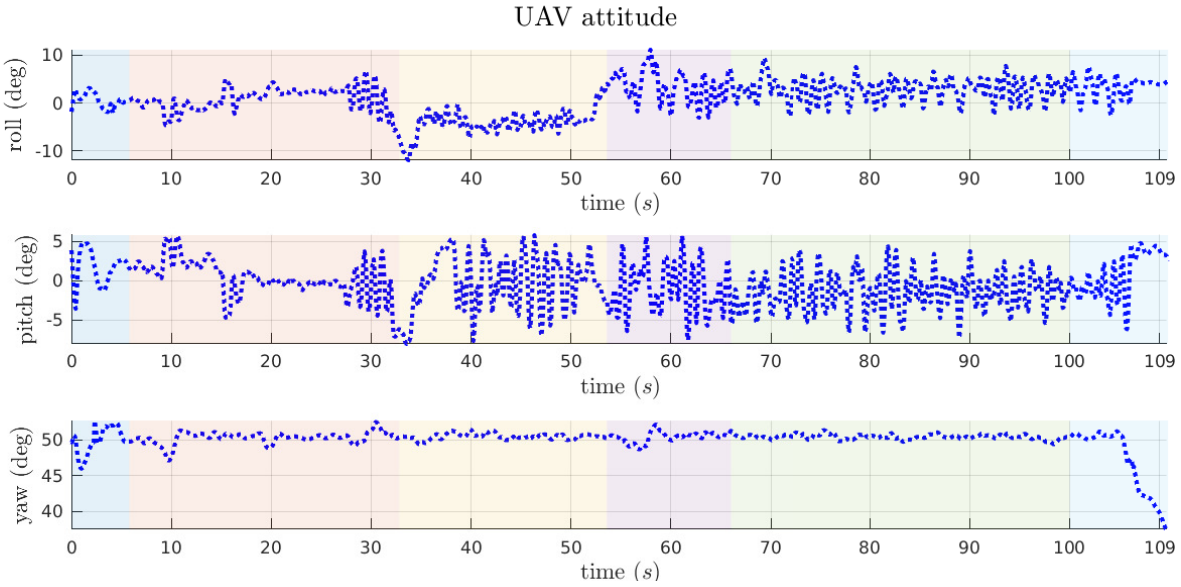


(b) USV's linear velocities in global coordinate frame.

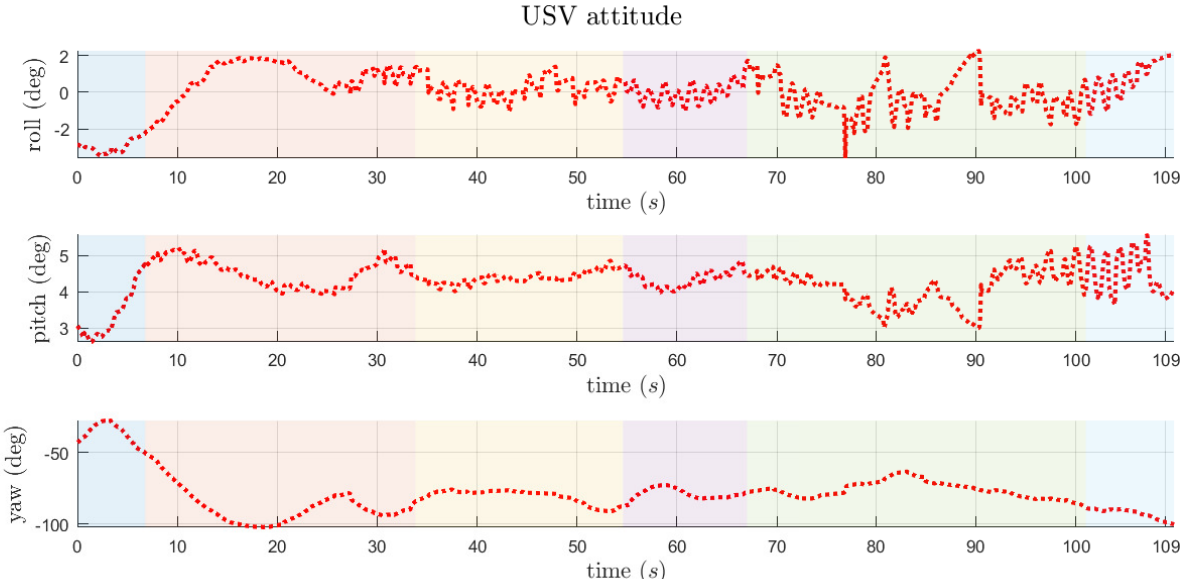
Figure B.3: USV's odometry. Roll was varying between ±2°, pitch angle was varying between ±4° and yaw was varying between −180° and 180°. The linear velocity in x-axis was varying between $-2.73\,\mathrm{m\,s^{-1}}$ and $0\,\mathrm{m\,s^{-1}}$, velocity in y-axis was varying between $-0.64\,\mathrm{m\,s^{-1}}$ and $2.28\,\mathrm{m\,s^{-1}}$ and vertical velocity was between $-1.26\,\mathrm{m\,s^{-1}}$ and $1.08\,\mathrm{m\,s^{-1}}$. The meaning of the background colours is following — dark blue: idle, orange: get height, yellow: approach, purple: tracking, green: tracking stable, light blue: landing and light red: flare.

Appendix C

Real-world experiment



(a) UAV's roll, pitch and yaw angles.



(b) USV's roll, pitch and yaw angles.

Figure C.1: UAV's attitude angles Fig. C.1a and USV's estimated attitude angles Fig. C.1b. The colour marks individual modes. The meaning of the background colours is following — dark blue: take-off, orange: get height, yellow: approach, purple: descent for tracking, green: tracking USV and light blue: landing.

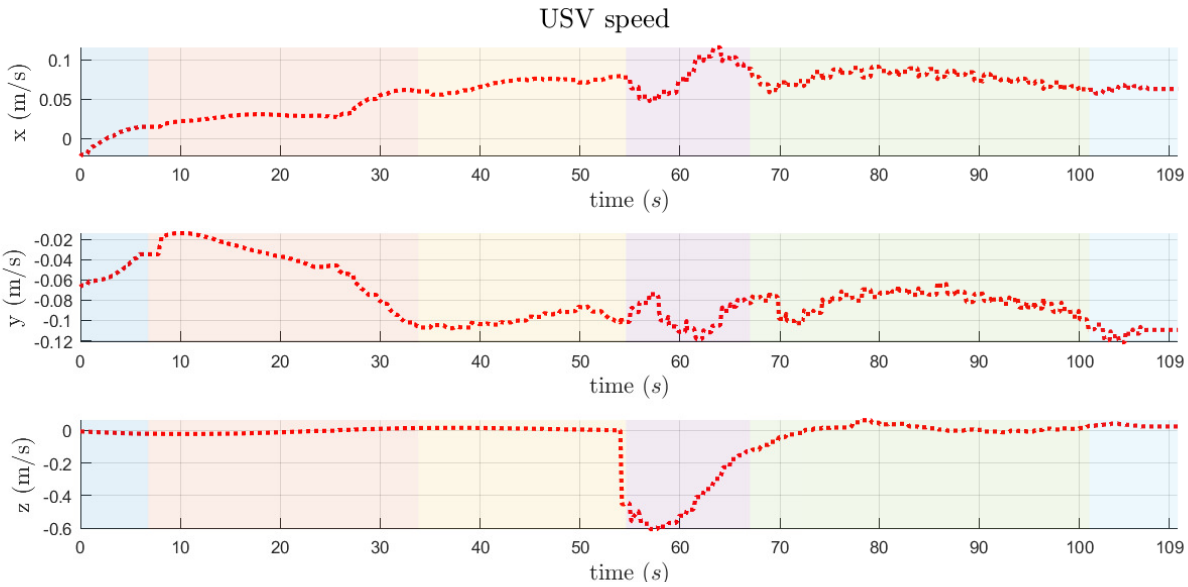


Figure C.2: USV's linear velocities in global coordinate frame. The meaning of the background colours is following — dark blue: take-off, orange: get height, yellow: approach, purple: descent for tracking, green: tracking USV and light blue: landing.

Appendix D

CD Content

In Table D.1 are listed names of all root directories on CD.

Directory name	**Description**
thesis	the thesis in pdf format
thesis_sources	LaTeX source codes
matlab	MATLAB source codes
repository	software source codes
videos	videos from real-world experiments

Table D.1: CD Content