Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Sondre Sortland

Autonomous contact-based thickness measurement from a multirotor UAV

Master's thesis in Cybernetics and Robotics
Supervisor: Tor Arne Johansen
June 2019

NTNU
Norwegian University of Science and Technology

scout
DRONE INSPECTION

Sondre Sortland

Autonomous contact-based thickness measurement from a multirotor UAV

Master's thesis in Cybernetics and Robotics
Supervisor: Tor Arne Johansen
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Faculty of Information Technology
and Electrical Engineering
Department of Engineering Cybernetics

MSC THESIS DESCRIPTION SHEET

Name: Sondre Sortland
Department: Engineering Cybernetics
Thesis title: Autonomous contact-based thickness measurement from a multirotor UAV

Thesis Description:
A key point in inspection of marine infrastructure and tanks is steel thickness measurements. Today, inspectors must climb or build time-consuming scaffolding to reach high and other hard-to-reach locations.

Recent advancement in sensor technology has enabled small, light-weigh ultrasonic thickness measurement gauges that can be mounted on small multirotor UAVs, that require only seconds of contact to take reliable measurements.

The thesis aims to investigate mounting such a sensor on an arm in front of the UAV, and develop models and control algorithms for automatic measurements.

The project is conducted in cooperation with Scout Drone Inspection AS, and the student will sign the NTNU "Standardavtale" with the company.

The following items should be considered:

1. Perform a literature review on ultrasonic thickness measurement performed from UAVs
2. Conduct simplified mathematical modelling of the interactions between the wall and the UAV, and investigate methods to model the constraints between the wall and UAV.
3. Design and review control algorithm strategies for automatic measurements, given an initial drone position some distance from the wall. Incorporate information about available local navigation systems.
4. Investigate and review techniques for local navigation (heading, distance) using on-board sensors.
5. Implement and test the automated test procedure on a test-drone supplied by Scout DI. The test should include:
    a. Interface and drivers to the necessary navigation and UTM measurement sensors.
    b. A one-press button to initiate the measurement maneuver, when the drone is steady an unknown distance from the wall.
    c. The drone shall then pe rform the measurement maneuver automatically and return to the initial distance from the wall after the maneuver.
    d. The drone should automatically detect (and possibly recover) from misalignments during the maneuver.
    e. After the maneuver is complete, the result of the thickness measurement is presented.
6. Conclude findings in a report.

Start date: 2019-01-07
Due date: 2019-06-03

Thesis performed at: Department of Engineering Cybernetics, NTNU
Supervisor: Professor Tor Arne Johansen, Dept. of Eng. Cybernetics, NTNU
Co-Supervisor: Dr. Kristian Klausen, Scout Drone Inspection AS

Abstract

This thesis investigates the automation of thickness measurements using an ultrasonic probe mounted on a drone. The use of unmanned aerial vehicles (UAVs) for non-destructive testing (NDT) in industrial environments have great potential for time and cost savings. Great success has been demonstrated in the area of visual inspection using drones, and is moving towards fully autonomous operations. Recent advancements in sensor technology has enabled small, lightweight ultrasonic thickness measurement gauges that can be mounted on small multirotor UAVs. These sensors require only seconds of contact to take reliable measurements, opening for the opportunity to do thickness measurements from multirotor UAVs.

This thesis has developed models and control algorithms for contact based interactions, inspired by related work on the topic. These controllers have further been validated using a high performance physics engine. The controllers are based on theory from force control, utilizing the impedance control framework, together with passivity based control. Furthermore, a method using tracking of pre-computed optimal trajectories have been developed and tested in the same physics engine setup. In addition, methods for local navigation based on data from a 2D scanning LIDAR using regression and RANSAC has been developed and tested.

Furthermore, complete scenarios with autonomous thickness measurements have been tested on multiple drone platforms for industrial inspection. In addition to the implementation of the impedance controllers mentioned above, this includes integration of an ultrasonic thickness measurement gauge, as well as sensors and methods for local navigation.

The results from both simulations in a physics engine and experimental validation shows that the controllers are capable of stable interaction with the environment, with solid robustness against disturbances. In conclusion, the controllers and navigation methods developed in this thesis, along with the integration carried out, have demonstrated a functioning solution for autonomous thickness measurements. This enables inspectors to carry out advanced inspection scenarios without prior experience or extensive pilot training.

Sammendrag

Denne avhandlingen undersøker automatiseringen av tykkelsesmålinger ved hjelp av en ultralydssonde montert på en drone. Bruk av ubemannede luftfartøyer (Unmanned Aerial Vehicle (UAV)) for ikke-destruktiv testing (Non Destructive Testing (NDT)) i industrielle miljøer har stort potensial for tid- og kostnadsbesparelser. Stor suksess er oppnådd innenfor visuelle inspeksjoner ved hjelp av droner, og nærmer seg helt autonome operasjoner. Nylige fremskritt innen sensorteknologi har gjort det mulig å montere små, lette ultralydsensorer for tykkelsesmåling på ubemannede multikopter. Disse sensorene krever kun sekunder med kontakt for å ta pålitelige målinger, og åpner for muligheten til å gjøre tykkelse målinger fra ubemannede multikopter.

Denne avhandlingen har utviklet modeller og kontrollalgoritmer for kontaktbaserte operasjoner, inspirert av relatert arbeid på emnet. Disse kontrollerne har videre blitt validert ved hjelp av en høyytelses fysikkmotor. Kontrollerne er basert på teori fra kraftkontroll, og tar i bruk rammeverk for impedansbasert kontroll, samt passivitet. Videre er en metode for følging av forhåndsgenererte optimale baner blitt utviklet og testet i samme fysikkmotoroppsett. I tillegg til dette har metoder for lokal navigasjon basert på regresjon og RANSAC på data fra en 2D, roterende LIDAR blitt utviklet og testet.

Videre har komplette scenarier med autonome tykkelsesmålinger blitt testet på flere droneplattformer for industriell inspeksjon. I tillegg til implementeringen av impedanskontrollene nevnt ovenfor inkluderer dette integrering av en ultralydtykkelsesmåler, samt sensorer og metoder for lokal navigasjon.

Resultatene fra både simuleringer i fysikkmotor og utførte eksperimenter viser at kontrollerne opnnår stabil kontakt med omgivelsene, med solid robusthet mot forstyrrelser. Til slutt har kontrollerne og navigasjonsmetodene utviklet i denne oppgaven, sammen med den utførte integrasjonen, vist en fungerende løsning for autonome tykkelsesmålinger. Dette gjør det mulig for inspektører å gjennomføre avanserte inspeksjonsscenarier uten tidligere erfaring eller omfattende pilotopplæring.

Preface

The work conducted during this thesis has been carried out at the Department of Engineering Cybernetics, in collaboration with Scout Drone Inspection AS (abbreviated Scout DI in this thesis). Scout DI has provided the necessary hardware to conduct the experiments in this thesis, along with guidance and introduction to the software systems used along with this hardware. Detailed information about what has been provided by Scout DI and what has been created as a part of this thesis will be presented in part V (Hardware, see introduction of chapter 14) and part VI (Software, see section 16.1). Scout DI has also provided facilities to conduct the experiments.

Contributions and related work are presented in the introduction.

This thesis is the continuation of a specialization project carried out in the fall of 2018. This was an exploratory simulation-study of the contact based inspection scenario, and includes the initial development of the controllers used in this thesis. Since the reader cannot be expected to have read the report from the specialization project, some parts have been adapted and included from this earlier work. Note that this is common practice at NTNU, but not commonly mentioned in theses. This applies to the parts about modeling, control and simulation (see outline in section 1.4). However, these parts have also been subject to improvements, revisions and further work during this thesis, along with new implementations into the software frameworks used on the drone hardware.

All illustrations and figures in this thesis are created by the author, unless explicitly stated otherwise.

Sondre Sortland
Trondheim, June, 2019

Acknowledgments

I would like to thank my supervisor, Professor Tor Arne Johansen, for the support and guidance provided during this thesis. I would also like to thank everyone involved with Scout Drone Inspection AS for allowing me to work on this thesis. A special thanks to Nicolai for the enthusiasm and encouragement throughout the process, Kristian for the continued guidance and fruitful discussions, Morten for his DUNE insight and intriguing C++ discussions, and Kristoffer for hardware guidance, pilot lessons and rendering of the image used on the cover of this thesis. Finally, I would also like to thank Arnt Erik Stene for his work on proof reading this thesis.

x

Contents

Acronyms

COG	Center of Gravity
COM	Computer on Module
DOF	Degrees of Freedom
DUNE	Unified Navigation Environment
ESC	Electronic Speed Control
GCS	Ground Control System
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
I2C	Inter-Integrated Circuit
IMC	Intermodule Communication
IMU	Inertial Measurement Unit
IR	infrared
LIDAR	Light Detection And Ranging
MAV	Micro Air Vehicle
MAVLink	Micro Air Vehicle Link
MEMS	Micro Electromechanical Systems
MPC	Model predictive control
NDT	Non Destructive Testing
NED	North-East-Down
NLP	Nonlinear Program

ODE	Open Dynamics Engine
QP	Quadratic Programming
RADAR	RAdio Detection And Ranging
RANSAC	Random Sample Consensus
ROS	Robot Operating System
SDF	Simulation Description Format
SLAM	Simultaneous Location And Mapping
SPI	Serial Peripheral Interface
ToF	Time of Flight
UART	Universal Asynchronous Receiver-Transmitter
UAV	Unmanned Aerial Vehicle
URDF	Unified Robot Description Format
USB	Universal Serial Bus
UTM	Ultrasonic Thickness Measurement

Chapter 1

Introduction

This chapter will first present the motivation for the research, before briefly discussing related work. Finally, a list of main contributions are presented, as well as an outline of the remainder of the thesis.

1.1 Background and Motivation

Industrial inspection is time intensive, costly and often imposes severe risk to the inspector. This holds true for inspection of cargo holds and tankers in the marine sector, where scaffolding is needed in order to reach all surfaces for inspection. A key point in inspection of marine infrastructure and tanks is steel thickness measurements. Mounting an ultrasonic thickness sensor on Unmanned Aerial Vehicles (UAVs) using a fixed end effector and developing models, control algorithms and navigation methods for reliable, automatic measurements, is a big step in the direction of making these inspections fully autonomous.

UAVs have a rich history in both research and engineering. The development of improved solutions for inertial sensors, satellite positioning and battery technology have a huge role in the shift from mainly military, to strongly increasing variety of civilian applications. Another strong contributor to this shift has been the development of lightweight and cheap Micro Electromechanical Systems (MEMS), for example accelerometers and gyroscopes.

Recent advancements in sensor technology have enabled small, lightweight ultrasonic thickness measurement gauges that can be mounted on small multirotor

UAVs. These require only seconds of contact to take reliable measurements, and a few also include mechanisms for automatic application of gel to the probe, in order to ensure better contact and more reliable measurements.

Figure 1.1: Inspection drone with camera. (Image courtesy of Scout DI)

Since contact with physical objects pose a huge risk for UAVs, control algorithms for safe interaction with the environment is needed. An extensive amount of work has been done in the field of force and contact based control for robots with fixed base. These robots have the benefit of being able to dissipate contact forces through the fixed base link. One particular challenge for most UAVs, including quadrotors, helicopters or ducted fans, are that these systems are mechanically underactuated, meaning that not all of their Degrees of Freedom (DOF) can be simultaneously controlled. This affects the control law design, as stability needs to be preserved in the presence of disturbances from physical interaction. This thesis will investigate controllers for stable interaction, enabling thickness measurements using drones.

Most drones available for commercial use require some sort of Global Navigation Satellite Systems (GNSS), such as the Global Positioning System (GPS) to solve the drift problem in dead reckoning in inertial navigation. Another common approach to the problem is the use of visual navigation, such as visual odometry based on camera or an optical flow sensor. However, neither of these are available in most industrial environments, as the weak GNSS signals fail to penetrate most building surfaces, and poor lighting render visual techniques useless in most situations. Hence, this thesis will also investigate local navigation solutions based on sensors that are able to operate in such environments, in order to

successfully carry out contact based inspections.

To harness the potential for time and cost savings, an end-to-end system for autonomous thickness measurements needs to be both robust and user-friendly. This puts strong requirements on the quality and security of the system. Drone operators with little to no prior experience or training should be able to carry out the same operations as an experienced drone operator could perform manually. A visualization of the automatic thickness measurement operation can be seen in fig. 1.2. This thesis will develop such a system by integrating automatic thickness measurements into an industrial drone platform, using the control and navigation solutions developed, along with operator interfaces for easy access.

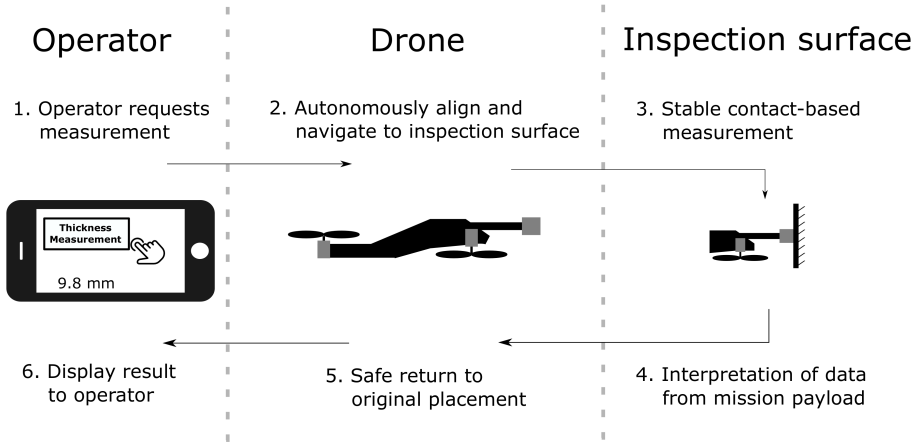


Figure 1.2: The inspection scenario

It should be noted that the meaning of the word autonomous in this context is not intended as a fully autonomous operation from takeoff to landing. As many industries have discovered in recent years, it is more beneficial to view autonomy as different levels, each taking a step towards the goal of fully autonomous operations. Hence, in this thesis, the operation will be autonomous from the moment the operator request a thickness measurement at a surface of interest, until the drone has returned safely to the original position. The methods developed will also focus on facilitating for future, fully autonomous inspection scenarios.

1.2 Related Work

In this section, the most relevant work for this thesis is presented. Note that the literature directly applicable to this thesis, will be covered in greater detail in the background chapters in their respective parts.

Automated inspections in general is a topic that has been researched thoroughly, mainly because of the obvious potentials for time and cost savings. An area that has received a lot of attention, is the use of drones to conduct visual inspections. Inspection of wind turbines is among the applications for visual inspections, and [66] proposes a method for visual navigation, while [64] presents a concept based on priori 3D mapping and spline-based flight path planning. In [73], Zhang et al. uses a photogrammetry payload to provide a 3D visual reconstruction of the blade profile, while [36] presents a feasibility study for defect detection using infrared thermography.

In recent years, visual inspections of buildings using drones have also been heavily researched. In [23], Eschmann et al. developed a rotary wing octocopter Micro Air Vehicle (MAV) system to scan buildings using a high resolution camera. A possibility study on building inspection is presented in [47], using a range of different sensors that are mounted on a drone, while [56] created motion corrected images used to form a facade map of the building. In [24], a complete system for use in indoor visual inspection is presented. Localization, mapping and navigation is achieved using only the embedded sensors and with the perception and control loop running on-board the MAV.

The EU funded R&D project MINOAS [14, 10, 55] is specifically concerned with the development of solutions for autonomous inspection of marine vessels. Their approach combines a drone for visual inspection [13] and a robotic crawler [21] for contact based inspections. Among the contributions are a mosaic approach for stitching images [37] for visual inspection and vision-based corrosion detection [54].

The AEROWORKS[1] project aims to develop drones equipped with robotic arms for advanced manipulation capabilities. Most relevant to this thesis is an LQR controller for substantial and sustained force introduced in [71]; however,

[1] http://www.aeroworks2020.eu/

it assumes that the drone is equipped with an actuator that is able to move the end effector to help counteract disturbances. Other control research under this project, summarized in [4], is mainly directed towards the use of a robotic manipulator arm mounted on a UAV.

The AEROARMS [5] project (EU funded R&D project) has tested a large drone conducting thickness measurements. The drone has 6 tilted rotors, making it able to produce forces in all directions and overcome the underactuation problem. The sensor arm is also rotatable, making it easier to conduct measurements at different angles. Even more recently, more work has been done on control approaches for fully actuated systems. A cascaded PID controller in free flight, while switching to an angular rate stabilizing control in contacts is presented in [72]. In [67], a PD control law for an elastic jointed manipulator model is used, and introduces integral effects in all directions except along the contact axis. Finally, [11] presents a selective impedance control strategy for omni-directional platforms.

More related to this thesis, work has also been done in the area of physical contact by rigid contact mechanisms. In the papers [2] and [3], a hybrid MPC solution, which enables a drone to draw on surfaces, is presented. The main drawbacks of this method is the complexity and the computation power required for the solution of the MPC problem. Drones utilizing horizontal propellers to achieve contact with the environment have been developed, such as in [1].

In [41] the mechanical design of a drone endowed with a small manipulator is developed. The drone generates horizontal forces by changing its attitude, similar to what is necessary to achieve the goals of this thesis. In [32], Fumagalli et al. proposes a control law based on the simplified 2D model of the drone, while [33] proposes a control law based on the full 3D model. In [65] a hybrid impedance and force controller is proposed for the same drone. While these papers use a stabilizing manipulator structure mounted on the drone, the theory is very relevant to this thesis and forms the basis for the control derivations in part III.

1.3 Contributions and Scope of This Thesis

The objective of this work is the design and implementation of control structures enabling automatic thickness measurements using an ultrasonic probe, in addition to navigation solutions in industrial environments based on available sensors. Integration of these methods and necessary sensors will together form a complete system for automatic thickness measurements. This thesis is the continuation of a simulation study carried out as part of the specialization project in the fall of 2018. Hence, the following contributions are partitioned based on which project they primarily can be attributed to. However, it should be noted that the elements from specialization project have been subject to improvements, revisions and further work in this thesis.

Specialization Project

- Dynamic models in both 2D and 3D, along with impedance based controllers for robust interaction with the environment, inspired by [32, 33].
- An exploratory derivation of a constraint model using the Udwadia Kalaba equations.
- Complete simulation scenario in Gazebo and ROS, and incorporating the UAV, measurement probe dynamics, inspection surface and interactions between the aforementioned parts.
- Implementation and validation of the impedance based control algorithms in ROS and Gazebo.
- A framework for control implementations in DUNE with simulations in ROS and Gazebo.

This Thesis

- Derivation of pre-computed trajectories for interaction in the framework of optimal control.
- Implementation and validation of pre-computed, optimal trajectories in ROS and Gazebo.
- Derivation and verification of algorithms for estimation of relative heading based on both RANSAC and regression.
- Experimental validation of impedance based control algorithms on multiple hardware platforms.
- Experimental validation of the developed methods for local navigation.
- Integration and verification of a complete, robust and user-friendly system for autonomous thickness measurements in an industrial context.

1.4 Outline and Organization

This thesis is split into several parts, categorized by important aspects for successful development of automatic thickness measurements. Modeling is concerned with finding models for control synthesis and basic simulations, but also more accurate models that can provide insight into key aspects of the interaction. In the control part, different approaches for stable control of the interaction will be presented. Thereafter, algorithms and methods for gathering necessary navigation information about the environment is presented. The two following parts will then describe the more practical sides of implementing the platform used in this thesis, and will detail the hardware and software used respectively. Finally, the simulation part is included in the appendix, which outlines the development of a framework for accurately simulating contact-based inspection from UAV in a physics engine. This simulation framework provides capabilities of thorough and realistic evaluation of a wide range of controllers before they are tested experimentally. Hence, ordered chronologically, this belongs before the experimental results from the flight tests, but has been moved to the appendix in order to give more room for experimental results in this thesis. However, the interested reader is highly encourage to also look into this part.

Seen from a broader perspective the modeling, control and navigation parts will provide the theoretical foundation of this thesis, while the software and hardware parts will provide information about the implementation on an industrial drone platform. The result and simulation parts will present the achieved results, and will give a thorough discussion about strengths and weaknesses of the developed solutions and overall system.

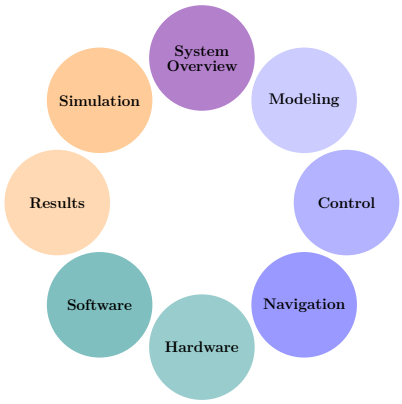


Figure 1.3: Overview of the main parts of the thesis

Detailed outline

Part I — System Overview:
Presents an overview of the drone and systems used.. This includes a description of the ultrasonic thickness measurement sensor, along with the most important characteristics of the drone.

Part II — Modeling:
This part presents the 2D and 3D dynamics models for the quadrotor, before augmenting the models with interaction forces and implementing them in MATLAB/Simulink as a simulator. Also, general theory and an introductory example for the Udwadia Kalaba equations are presented, along with derivations for the constrained interaction during contact-based inspection.

Part III — Control:
First, a brief introduction to the general force control theory is presented. Secondly, impedance based control algorithms are derived based on the models developed in part II, before initial validation is carried out in the MATLAB/Simulink simulators. Lastly, an approach using optimal trajectories are presented as an alternative to the impedance based controllers.

Part IV — Navigation:
This part will first present theory on RANSAC, before a brief discussion on navigation solutions based on available sensors. Thereafter, navigation methods based on data from a 2D scanning LIDAR using RANSAC and regression is presented.

Part V — Hardware:
In this part the hardware components of the UAV are presented, along with a short description of the integration process. This part will also detail the sensor and mission payloads used throughout this thesis.

Part VI — Software:
First, The LSTS Toolchain is presented, with special focus on the runtime environment DUNE. Thereafter, the development and integration of the different components described in this thesis is outlined, along with other components and application that were developed in order to create a complete solution for contact-based thickness measurement.

Part VII — Results:
In this part the results of the experimental validation are shown. Four different scenarios are shown, and the results are discussed and compared to results achieved in simulation.

Chapter 19 — Discussion and Conclusion:
An overall discussion of the achieved results is conducted and a conclusion is drawn with respect to the goals of this thesis. Finally, possible future work following this thesis is presented.

Appendix Part I — Simulation:
Background theory on the ROS and Gazebo framework is presented. Thereafter follows a full description of the setup and extensions made to the simulator to replicate the inspection scenario. Then, simulation results from different scenarios using impedance based control are presented and discussed. Finally, the simulation result of using pre-computed optimal trajectories is presented and discussed.

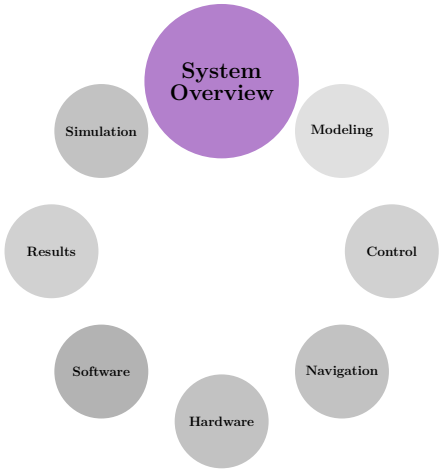
Notes about wording used in this thesis

The acronym UAV (Unmanned Aerial Vehicle) incorporates all types of unmanned aerial vehicles, including fixed-wing, helicopters and multirotors. Some ambiguity also applies to the word drone, which is the word most commonly used about quadrotors on the commercial market. However, in the following the words UAV and drone will be used interchangeably to mean a quadrotor, unless explicitly stated otherwise.

The word ”environment” is used to describe the inspection surfaces the drone interacts with. This is slightly ambiguous, as ”environment” usually incorporates much more than just the physical objects of the surroundings; however, it follows the phrasing used in the literature about physical interaction for aerial platforms, such as [32].

Part I

System Overview



Chapter 2

System Overview

This chapter gives a brief introduction to the drone characteristics, navigation sensors and the ultrasonic probe system. This is presented first, as the specifications are important for the modeling and control system synthesis in the following parts, but also to give the reader a more intuitive and visual understanding of the system.

2.1 The Development Drone

This thesis is conducted in collaboration with Scout Drone Inspection AS, who has designed the UAVs used throughout this thesis. The development drone is shown in fig. 2.1, with the probe mounted in front of the drone. An approximated set of parameters for mass and moments of inertia are given in table 2.1. These parameters will be the basis for all simulations created during this thesis. Figure 2.2 shows an exploded view, revealing all the components of the drone. The integration of hardware components will be detailed in part V.

Parameter	Value
M_{uav}	$2.5\,\mathrm{kg}$
I_{xx}	$0.01\,\mathrm{kg\,m^2}$
I_{yy}	$0.01\,\mathrm{kg\,m^2}$
I_{zz}	$0.02\,\mathrm{kg\,m^2}$

Table 2.1: Approximated drone parameters

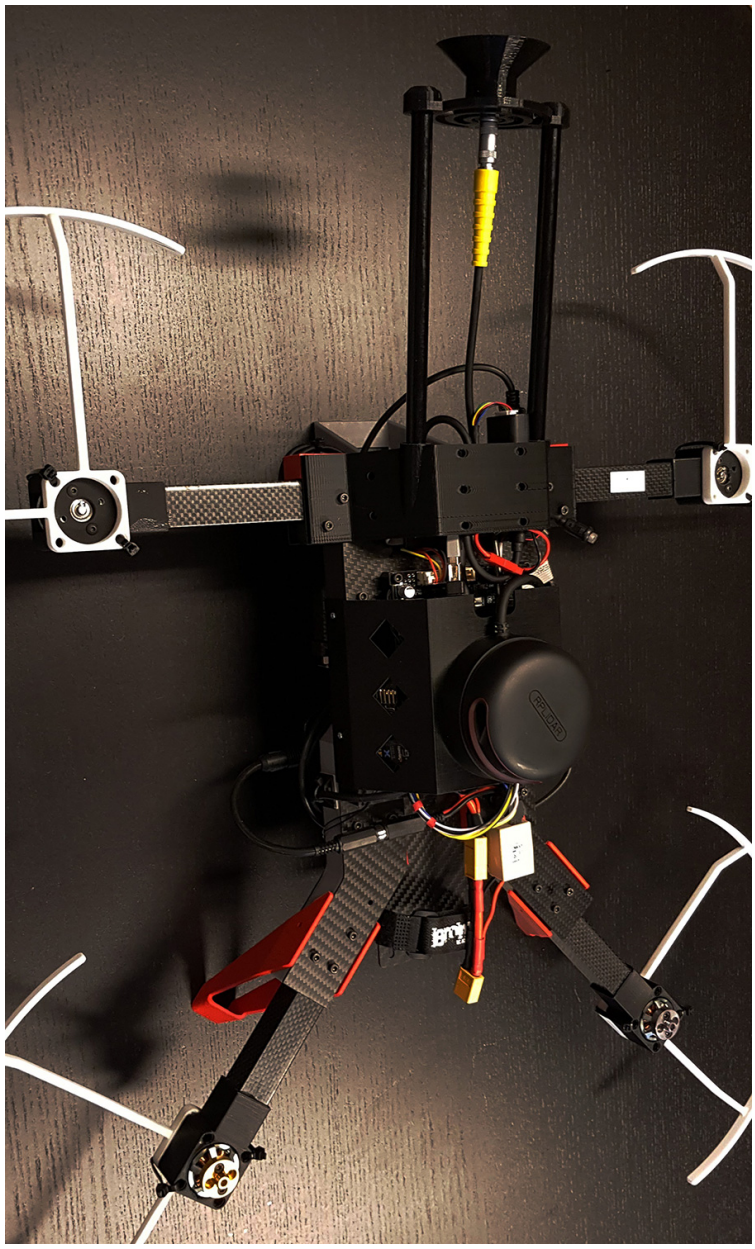


Figure 2.1: Fully assembled drone with all sensors and mission payload

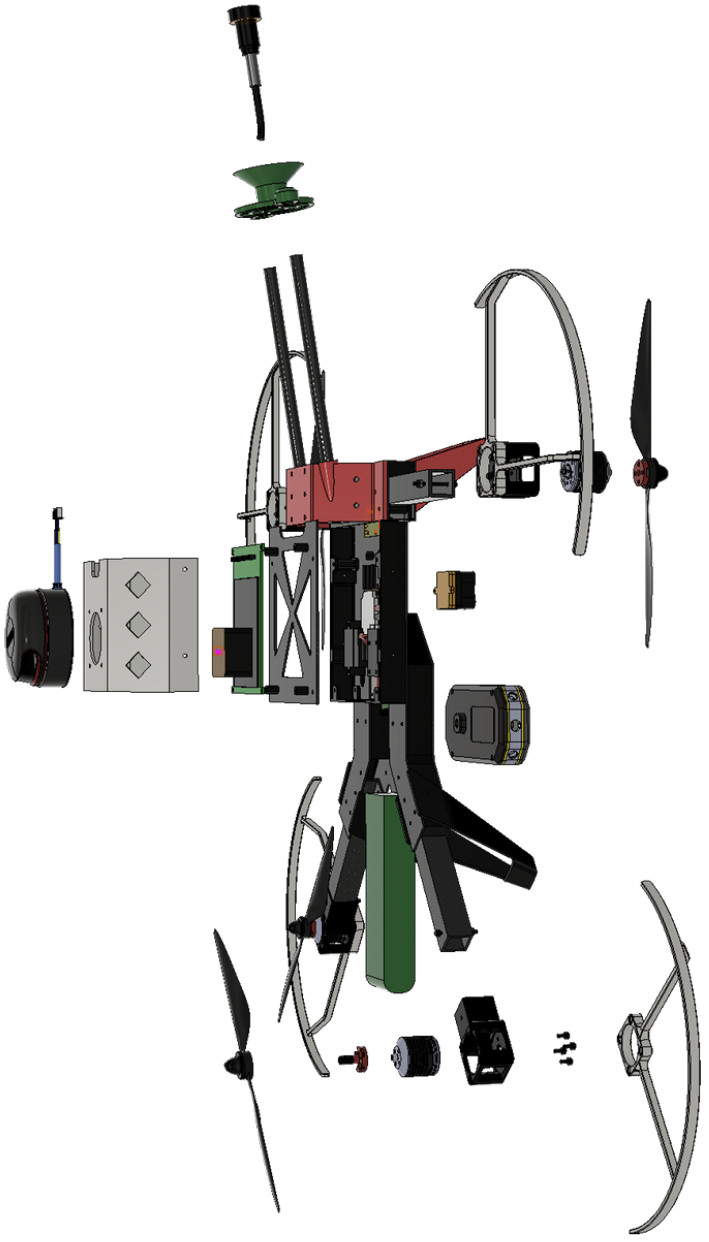


Figure 2.2: Exploded view of the drone and all hardware components. (Courtesy of Scout DI)

2.2 Sensor Overview

This section presents an overview of the most important navigation sensor and mission payload on board the drone. This includes a Light Detection And Ranging (LIDAR) system for horizontal positioning and a range sensor for vertical positioning, as well as the mission payload used to conduct thickness measurements. Together, these provide the essential capabilities needed to carry out an inspection scenario in an indoor, industrial environment.

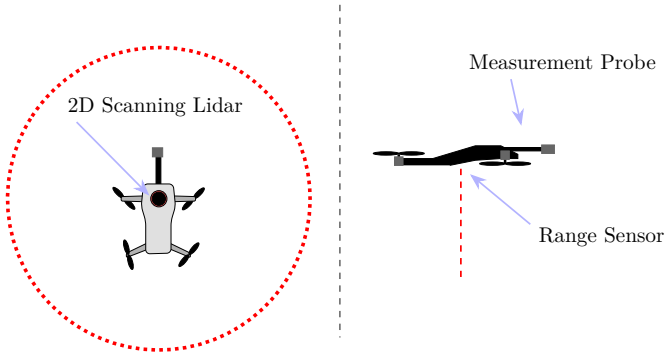


Figure 2.3: Overview of the most important navigation sensors and mission payload

2.2.1 Navigation Sensors

LIDAR

The main navigation sensor used for positioning and orientation in the xy-plane is a 2D scanning LIDAR. The LIDAR is mounted on top of the drone (as seen in fig. 2.2 and fig. 2.1). It provides a laser scan of the environment aligned with the body xy-plane, as illustrated on the left side in fig. 2.3.

Navigation solutions for indoor environments based on a 2D scanning LIDAR are developed in part IV, while more details about the sensor and software integration will be given in section 14.6 and section 16.2 respectively.

Range Sensor

The drone features a Time of Flight (ToF) range sensor to measure the altitude above ground. The sensor is mounted to the bottom side of the drone, and can be seen in black and yellow in the lower part of fig. 2.2. More details on the sensor specification and integration will be given in section 14.3. Combining the scanning LIDAR aligned with the xy-plane, and the range sensor pointing along the z-axis, this gives enough information to do 3D position estimation in poorly lit, GPS denied environments.

2.2.2 Steel Thickness Measurement

In this thesis, Ultrasonic Thickness Measurement (UTM) is carried out using a probe made by Tritex NDT[1], which is especially designed to be mounted on drones. It features a probe and a small box used for data processing and collection, which can be seen in fig. 2.4.

The mounting of the probe onto the hardware platform is described in section 14.5, while section 16.2 focuses on the integration into the software framework used on the drone. In addition, section 14.5 will also give an introduction to thickness measurements using ultrasound, and describe the theory behind the measurement technique used in the probe.

Figure 2.4: The UTM probe. Image courtesy of Tritex NDT

[1] https://www.tritexndt.com/

The probe system includes a mechanical compliance device to help the drone interact with the environment. The compliance device depicted in fig. 2.4 is included in the standard product. However, introductory experiments also showed that in order to maximize the probability of a valid measurement, a small angle between the probe and surface was needed. The mechanical compliance system is designed to compensate for this by having a flexible joint near the end of the probe. In order to get the desired disturbance rejection, a custom probe mounting was developed by Scout DI (seen in both fig. 2.1 and fig. 2.2). Such mechanical compliance devices have also been included in the simulations developed in this thesis, in order to ensure as precise replication of the physical system as possible.

The control system developed focuses on creating a stable interaction, while at the same time providing sufficient contact forces and minimize any orientation errors in order to get a valid measurement from the probe. In part VII, the results of this thesis is presented, and a thorough analysis is given with respect to these performance metrics.

2.3 Scout 135 Platform

The controllers and navigation methods developed are also tested on a another drone platform developed by Scout DI. These experiments are carried out to show that the solutions are hardware agnostic, but also give more detailed insight into how the hardware design affect the performance of the control systems.

The drone can be seen in fig. 2.5, and a detailed description of the differences between the two platforms will be given in section 14.7.

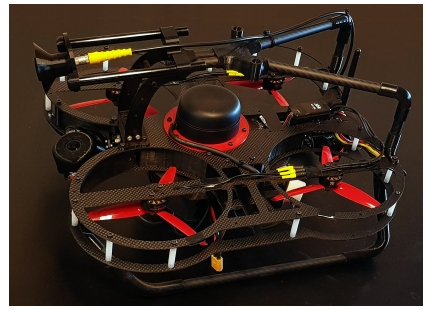
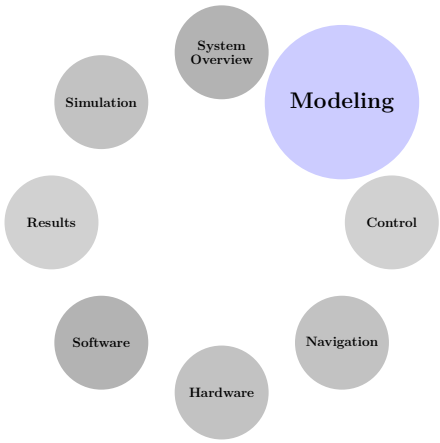


Figure 2.5: The Scout 135 platform

Part II

Modeling



Chapter 3

Introduction

As this thesis will present multiple different models for the same scenario, this chapter will briefly give the motivation and reasoning behind this, their use-cases and how they fit in with the rest of this thesis.

Figure 3.1 illustrates how different models can fit into the same control scenario, and shows a typical ecosystem for developing models and controllers for a general application. An accurate and advanced model can usually be derived based on the physical system, and forms the basis for an precise simulation of the system. These models usually encompass all of the detailed physical behavior of a system; however, they are in most cases not suitable for control synthesis. In fact, it might not even be feasible as the model might be implicit, too complex to analyze or appear as a black box from a control perspective (such as data driven models). In order to create a model more suitable for control, simplifications are made, such as linearization or reduction of the model.

In this part, two models for control synthesis will be presented: One reduced model in two dimensions and one 3D model. Simulators are also developed in MATLAB/Simulink to do an initial verification of the control approach. Another model, based on the Udwadia Kalaba equations, are also presented in chapter 6; however, not for the purpose of control synthesis, but rather as a more accurate model that might be used to build simulators in software such as MATLAB/Simulink.

Based on the simplified design models, controllers can be derived. It is important to emphasize the possibility to use the same model to develop different controllers

by utilizing other control techniques. Part III will present control approaches based on the models derived in this part. Afterwards, these will be tested in MATLAB/Simulink simulators, as an initial validation of the approach.

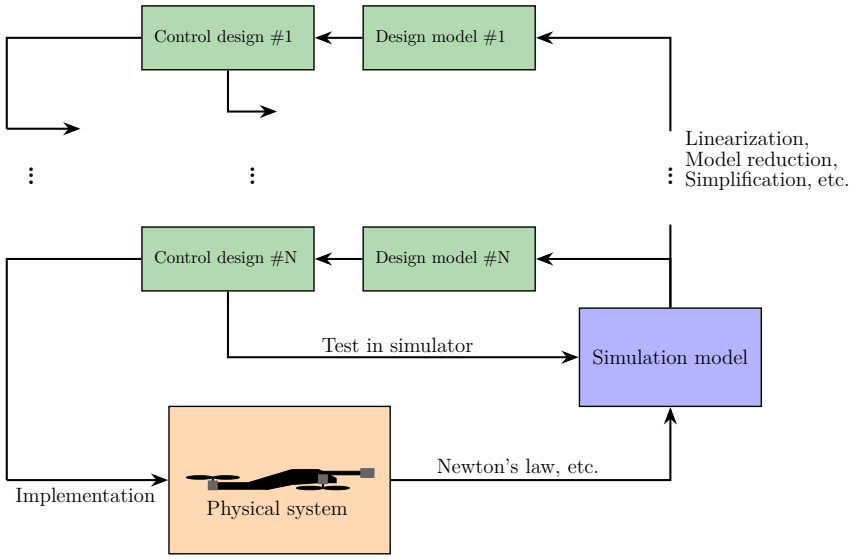


Figure 3.1: Ecosystem of model designs for control

After the initial verification of the controller, a more thorough verification can be done using the advanced simulation model. In this thesis, this is done using a physics engine. This usually gives a more realistic scenario compared to the models used for control design, as it includes more of the key characteristics, behaviors and functions of the physical system. In the ideal case this will represent an almost identical case as the physical system, and only small adjustments are needed to implement the controller on the physical system. Verifications on the simulation model will be carried out in part Appendix Part I, when the system and controller are implemented in ROS/Gazebo.

Finally, the developed controllers can be tested on the physical system itself. In this thesis these results are presented in part VII. In section 18.5 a comparison is made between the simulated scenarios and the flight tests, highlighting the key similarities and differences between them, both in terms of the modeling accuracy and controller performance.

Chapter 4

2D UAV Model

This chapter will introduce an approximated 2D dynamical model for the drone and then augment the model for planar interaction to include contact forces. After that, simulations are performed to show the validity of the model. A lot can be learned about this scenario using only a simplified 2D model as a representation, which will be used to get experience and insight before the full 3D model is derived. Also, the model will be used for control synthesis later. The following derivations adopts the conventions used in most of the literature, such as [29].

4.1 2D Model Basics

The 2D model considers a planer view of the scenario, as seen in fig. 4.1. This section will very briefly cover the basics of a simple 2D model for a drone.

The coordinate frames are given in fig. 4.1 and follows the NED coordinate frame convention (the y-axis completes the right-hand rule and points outwards in this example). NED is also assumed inertial.

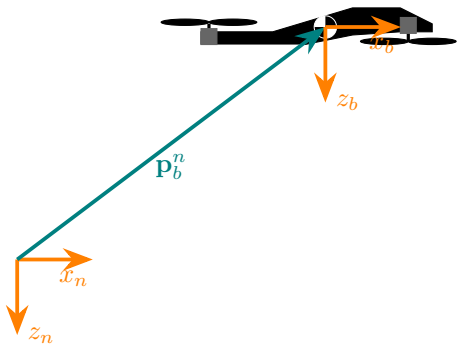


Figure 4.1: 2D model of the drone

Symbol	Description
$f_i, \quad i \in \{1,2,3,4\}$	Magnitude of motor thrusts (enumerated clockwise from front left motor)
L_f	Lever arm front
L_b	Lever arm back
f^b	Resulting thrust from rotors in body frame
τ^b	Resulting torque from rotors in body frame

Table 4.1: A definition of the symbols used in the 2D model

Consider a UAV with lever arm L_b for the two back propellers and L_f for the front propellers. Using the variables described in table 4.1, f^b and τ^b are then given by

$$f^b = f_1 + f_2 + f_3 + f_4 \tag{4.1}$$

$$\tau^b = L_f f_1 + L_f f_2 - L_b f_3 - L_b f_4 \tag{4.2}$$

Which can be represented in a more convenient vector-form as

$$\begin{bmatrix} f^b \\ \tau^b \end{bmatrix} = \begin{bmatrix} -1 & -1 & -1 & -1 \\ L_f & L_f & -L_b & -L_b \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{4.3}$$

In the case of the planer dynamics where only the pitch dynamics are considered, it does not matter which of the rotors in the front pair or back pair that produces

the force. Hence, given f^b and τ^b, this set of equations can be solved for all f_i by constraining $f_1 = f_2$ and $f_3 = f_4$.

Let $\mathbf{f}_e = \begin{bmatrix} f_{e,x} & f_{e,y} & \tau_e \end{bmatrix}^\top$ be any interaction from the environment. The unconstrained dynamics can be written as [32]:

$$\begin{aligned} M_{uav}\ddot{x} &= f^b \sin\theta + f_{e,x} \\ M_{uav}\ddot{z} &= f^b \cos\theta + M_{uav}g + f_{e,y} \\ J_{uav}\ddot{\theta} &= \tau^b + \tau_e \end{aligned} \tag{4.4}$$

Or in vector form

$$\underbrace{\begin{bmatrix} M_{uav} & 0 & 0 \\ 0 & M_{uav} & 0 \\ 0 & 0 & J_{uav} \end{bmatrix}}_{\mathbf{M}_{uav}} \underbrace{\begin{bmatrix} \ddot{x} \\ \ddot{z} \\ \ddot{\theta} \end{bmatrix}}_{\ddot{\mathbf{x}}} = \underbrace{\begin{bmatrix} \sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{R}(\mathbf{x})} \underbrace{\begin{bmatrix} f^b \\ \tau^b \end{bmatrix}}_{\mathbf{u}} + \underbrace{\begin{bmatrix} 0 \\ M_{uav}g \\ 0 \end{bmatrix}}_{\mathbf{g}} + \mathbf{f}_e \tag{4.5}$$

Looking at the top two rows of $\mathbf{R}(\mathbf{x})$, the underactuation of the system becomes apparent. The sub-matrix defined by the x and z-dynamics can never have full rank, and hence cannot be independently controlled. To summarize, the total system can be written as

$$\mathbf{M}_{uav}\ddot{\mathbf{x}} = \mathbf{R}(\mathbf{x})\mathbf{u} + \mathbf{g} + \mathbf{f}_e \tag{4.6}$$

Where $\mathbf{M}_{uav}$ is the mass-matrix of the system, $\mathbf{x} = \begin{bmatrix} x & z & \theta \end{bmatrix}^\top = \begin{bmatrix} \mathbf{p}_b^{n\top} & \theta \end{bmatrix}^\top$ is the state vector, $\mathbf{u}$ is the forces and moments from the rotors and $\mathbf{g}$ is the gravitational force.

4.2 Contact Force Augmentation

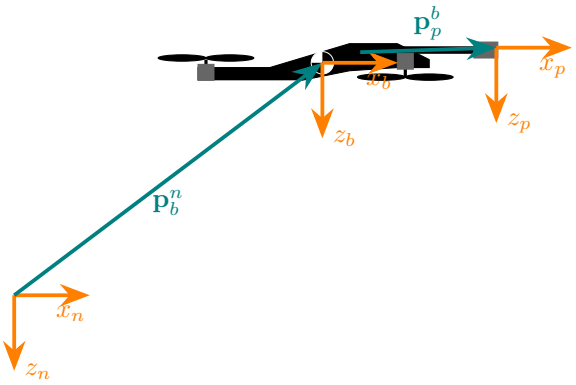


Figure 4.2: 2D model with contact

This section is inspired by [32], but differs with respect to the coordinate systems used, drone setup and the model of compliant behavior.

Equation (4.6) is now augmented with external forces and moments ($\mathbf{f}_e$), such that

$$\mathbf{M}_{uav}\ddot{\mathbf{x}} = \mathbf{R}(\mathbf{x})\mathbf{u} + \mathbf{g} + \mathbf{f}_e \tag{4.7}$$

The external forces can be found as [32]:

$$\mathbf{f}_e = \begin{bmatrix} -f_c \cos(\theta + \theta_p) \\ f_c \sin(\theta + \theta_p) \\ h f_c - (r - \eta) mg \end{bmatrix} \tag{4.8}$$

Where m is the mass of the sensor, θ_p is the angle between the probe and the x-axis, r is the length of the probe, η is the deformation of the compliance device, f_c is the contact force, assumed to work along the vector of the probe $\mathbf{p}_p^b$ and h is the lever arm for contact forces. One note is that the relationship between r,

η, θ_p, h and $\mathbf{p}_p^b = \begin{bmatrix} p_{p,x}^b & p_{p,y}^b \end{bmatrix}^\top$ is given by

$$\left\| \mathbf{p}_p^b \right\| = r - \eta \tag{4.9}$$

$$\theta_p = \text{atan2}\left(p_{p,y}^b, p_{p,x}^b\right) \tag{4.10}$$

$$h = \frac{p_{p,z}^b r_{p,x}^b - p_{p,x}^b r_{p,z}^b}{\left\| \mathbf{p}_p^b \right\|} \tag{4.11}$$

Where $\mathbf{r}_p^b = \begin{bmatrix} r_{p,x}^b & r_{p,y}^b \end{bmatrix}^\top$ is the vector from the origin to the point where the probe is attached in the body frame.

By assuming that the mass of the sensor is small compared to the mass of the UAV, such that $m << M_{uav}$, eq. (4.8) simplifies to

$$\mathbf{f}_e = \begin{bmatrix} -f_c \cos\left(\theta + \theta_p\right) \\ f_c \sin\left(\theta + \theta_p\right) \\ h f_c \end{bmatrix} = f_c \begin{bmatrix} -\cos\left(\theta + \theta_p\right) \\ \sin\left(\theta + \theta_p\right) \\ h \end{bmatrix} \tag{4.12}$$

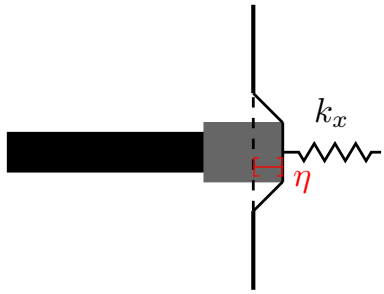


Figure 4.3: Interaction model

The contact force f_c is found by modeling the environment as a compliant surface such that the force is proportional to the deformation of the material [32]. This is equivalent to modeling the compliant behavior of the probe.

$$f_c := \begin{cases} -k_x \eta & \text{if } \eta > 0 \\ 0 & \text{if } \eta \leq 0 \end{cases} \tag{4.13}$$

where k_x is the collision stiffness and $\eta = x_p - x_w$ is the collision distance between the probe placement x_p and the wall placement x_w.

4.2.1 Compliant Behavior Considerations

An important note in this derivation, is that the interaction forces are modeled as directly affecting the dynamics of the drone. This is reasonable when the compliant behavior is simple, such as in this case. However, another possibility is to model the compliant behavior of the drone as a separate subsystem, such that the system becomes a cascaded interconnection of the environment, the compliance device, and the lateral/vertical dynamics of the vehicle. The environment would then influence the manipulator dynamics through the force f_c, and this propagates to the lateral and vertical dynamics of the quadrotor through a force $f_{compliant}$. This approach is more beneficial if the compliant device is more complex, and incorporates the scenario where a small manipulator is used as compliance device, such as in [32].

4.3 Simulation Validation

In this section simulations are carried out to verify that the model works as expected and forms the basis of a simulator for validation of the controllers. A low-level PD attitude controller and altitude control controller with gravity compensation was also used to enable verification of the interaction model in simulation.

In the following simulation the pitch reference is set to $-3°$ for 5 seconds, starting at $t = 10\,\text{s}$. This causes the UAV to accelerate towards the inspection surface placed at $x = 0.75\,\text{m}$. In fig. 4.4 it can be seen from the pitch angle θ that the contact causes the UAV to rotate backwards. The rotation along with the compliance forces from the interaction, results in a backward motion. Figure 4.5 shows the drone before, during and after the interaction and confirms this behavior. Afterward the first interaction, the attitude controller corrects the pitch and the drone accelerates towards the inspection surface again for another bounce. As the probe is angled slightly upwards during the interaction, it can also be seen that the drone is pushed down in the interaction. All of the described behavior is in accordance with physical interpretations, and the model seems to capture the most prominent effects of the interaction.

A side note is that these simulations needs to be run with a fixed step solver, as the performance of a variable step solver is sub-optimal due to the switching behavior of the contact forces in the interaction.

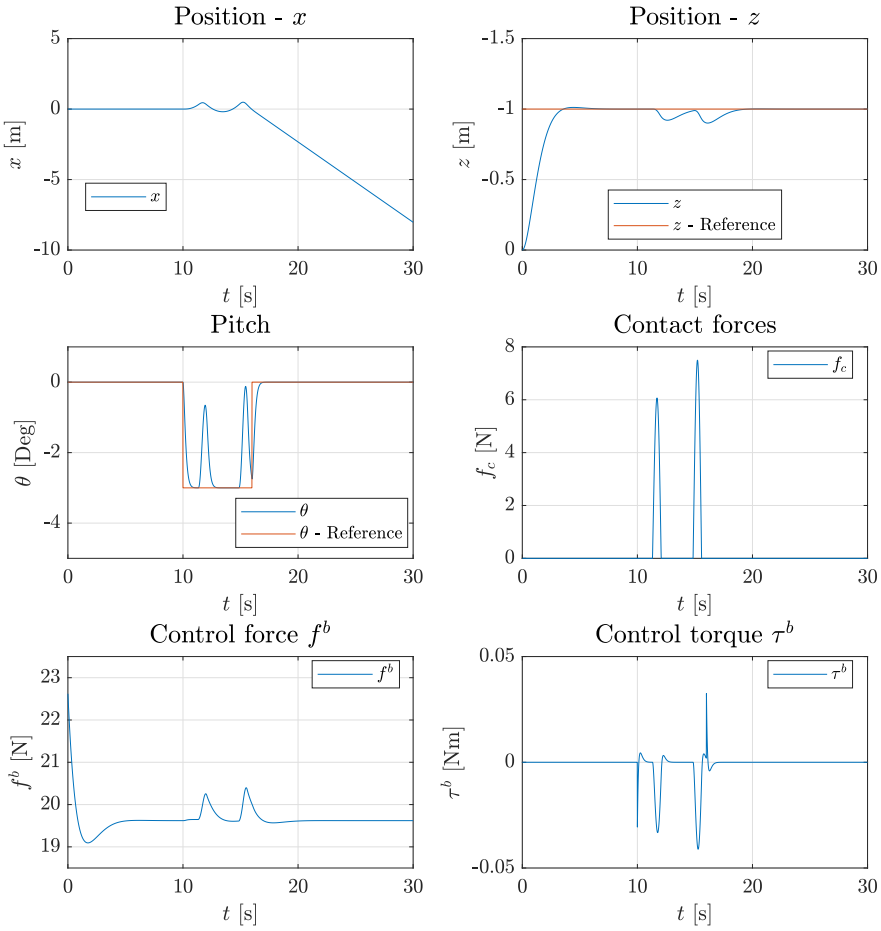


Figure 4.4: State variables, contact forces and control variables during simulation

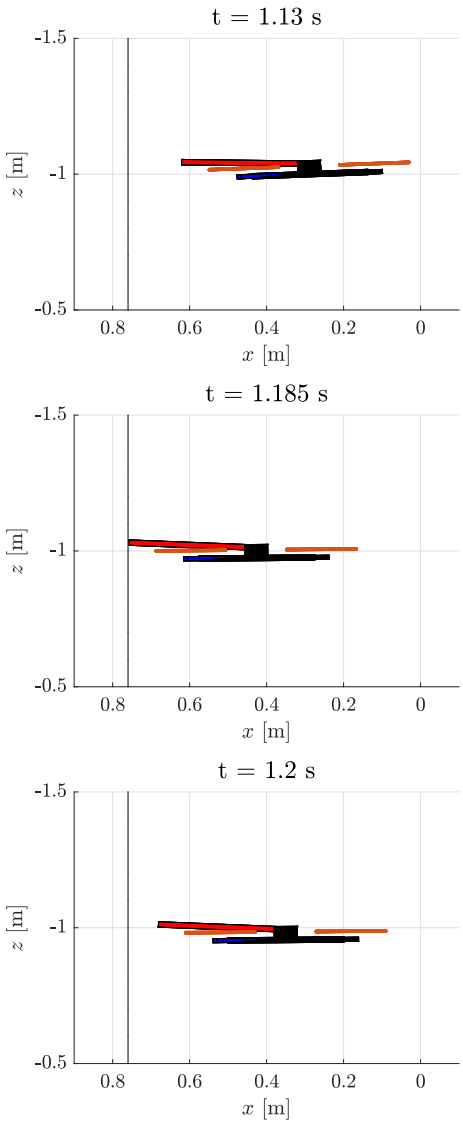


Figure 4.5: Planer view of the UAV before, during and after interaction

Chapter 5

3D UAV Model

This section will introduce a full 3D model for the drone and augment this model to include contact forces based on [33]. Thereafter, simple simulations are performed to validate the model. The model will also be used for control synthesis in section 9.1. The following derivations adopts the conventions used in most of the literature, for example [29].

5.1 3D Model Basics

This section covers the basics of modeling the dominant dynamics of a multirotor UAV, and is based on [29] and [43]. This is a topic that has been well studied in the literature, and hence this will only be a brief introduction. There are different choices of kinematic representation of attitude to be considered. Euler angles are commonly used for the easy interpretation and will be used throughout this thesis. However, other representations exists, such as quaternions, which has the advantage of being singularity free. As the main purpose of this thesis is developing a model for stable control, the singularity caused by the Euler-angle representation is easily avoided. As with the 2D model, the NED frame is assumed inertial in the following derivations.

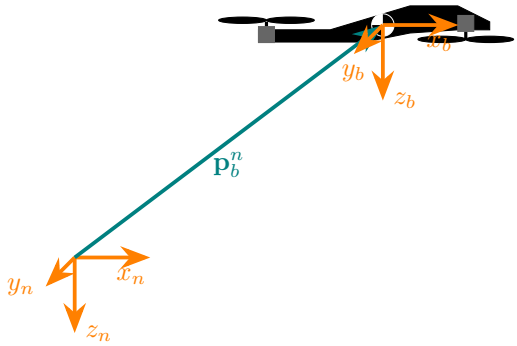


Figure 5.1: 3D model

5.1.1 Kinematics

Coordinates in the NED frame are given by $\boldsymbol{\eta} = \begin{bmatrix} x^n & y^n & z^n & \Theta^\top \end{bmatrix}^\top = \begin{bmatrix} \mathbf{p}_n^{b\top} & \Theta^\top \end{bmatrix}^\top \in \mathbb{R}^6$. Here $\Theta = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^\top$ are the Euler-angles defined by the zyx-convention [29]. The body-fixed coordinate system, $\{b\}$, is attached to the rigid body. The body-velocities is given by

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^\top \tag{5.1}$$

where $\boldsymbol{\nu}_1 = \begin{bmatrix} u & v & w \end{bmatrix}^\top$ is the translational velocities and $\boldsymbol{\nu}_2 = \begin{bmatrix} p & q & r \end{bmatrix}^\top$ are the rotational velocities.

It is convenient to define the rotation matrix [29], describing the rotation of the frame $\{b\}$ relative to a frame $\{n\}$ subject to the rotation $\boldsymbol{\Theta}$.

$$\mathbf{R}_b^n := \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}$$

where $c\cdot = \cos(\cdot)$ and $s\cdot = \sin(\cdot)$. Also, let the skew symmetric matrix [29] be defined as

$$\mathbf{S}(\lambda) := \begin{bmatrix} 0 & -\lambda_3 & \lambda_2 \\ \lambda_3 & 0 & -\lambda_1 \\ -\lambda_2 & \lambda_1 & 0 \end{bmatrix} \tag{5.2}$$

where $\lambda \in \mathbb{R}^3$, and $\mathbf{S}$ satisfies $\mathbf{S}(\lambda)^\top = -\mathbf{S}(\lambda)$ and $\mathbf{v}_1 \times \mathbf{v}_2 = \mathbf{S}(\mathbf{v}_1)\mathbf{v}_2$.

The transformation matrix $\mathbf{T}_\Theta$ [29], relates the angular velocities in $\{n\}$ and $\{b\}$ by $\dot{\Theta} = \mathbf{T}_\Theta \boldsymbol{\nu}_2$, and is defined as

$$\mathbf{T}_\Theta := \begin{bmatrix} 1 & s\phi s\theta/c\theta & c\phi s\theta/c\theta \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix}, \forall \theta \neq \frac{\pi}{2} + k\pi, k \in \mathbb{Z} \tag{5.3}$$

To conclude, the velocities in $\{n\}$ and $\{b\}$ are related by the equation

$$\dot{\eta} = \mathbf{J}_\Theta \boldsymbol{\nu} \tag{5.4}$$

where

$$\mathbf{J}_\Theta = \begin{bmatrix} \mathbf{R}_b^n(\Theta) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{T}_\Theta \end{bmatrix} \tag{5.5}$$

5.1.2 Kinetics

According to [29], the rigid body kinetics can be written as

$$m\left(\dot{\boldsymbol{\nu}}_1 + \boldsymbol{\nu}_2 \times \boldsymbol{\nu}_1\right) = \boldsymbol{\tau}_1 \tag{5.6}$$

$$\mathbf{I}_{CG}\dot{\boldsymbol{\nu}}_2 + \boldsymbol{\nu}_2 \times \left(\mathbf{I}_{CG}\boldsymbol{\nu}_2\right) = \boldsymbol{\tau}_2 \tag{5.7}$$

where m is the mass of the body, $\mathbf{I}_{CG} \in \mathbb{R}^{3\times 3}$ is the moment of inertia about the centre of gravity. $\boldsymbol{\tau}_1 \in \mathbb{R}^3$ and $\boldsymbol{\tau}_2 \in \mathbb{R}^3$ is external forces and external moments respectively. Under the assumption that $\{b\}$ is located in the center of gravity, then eqs. (5.6) and (5.7) can be written as

$$\mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}\boldsymbol{\nu} = \boldsymbol{\tau}_{RB} \tag{5.8}$$

where

$$\begin{aligned} \mathbf{M} &= \begin{bmatrix} m\mathbf{I}_{3\times 3} & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & \mathbf{I}_{CG} \end{bmatrix} \\ \mathbf{C} &= \begin{bmatrix} m\mathbf{S}\left(\nu_2\right) & \mathbf{0}_{3\times 3} \\ \mathbf{0}_{3\times 3} & -\mathbf{S}\left(\mathbf{I}_{CG}\nu_2\right) \end{bmatrix} \end{aligned} \tag{5.9}$$

The gravitational forces is easily defined in NED as $\mathbf{f}_G = \begin{bmatrix} 0 & 0 & mg \end{bmatrix}^\top$. Rotating them into body frame yields

$$\mathbf{g}^b = -\begin{bmatrix} \left(\mathbf{R}_b^n\right)^\top \mathbf{f}_G \\ \mathbf{0}_{3\times 1} \end{bmatrix} \tag{5.10}$$

To summarize, the kinematics and kinetics of the rigid body can be written as ([29])

$$\dot{\eta} = \mathbf{J}_\Theta \nu \tag{5.11}$$

$$\mathbf{M}\dot{\nu} + \mathbf{C}\nu + \mathbf{g}^b = \boldsymbol{\tau}_A \tag{5.12}$$

where $\boldsymbol{\tau}_A$ consist of all external forces, except for gravity.

5.1.3 Dominant Multirotor Dynamics

In the literature, such as [43, 33], it is common to leave the translational dynamics represented in $\{n\}$, rather than $\{b\}$. This is because a multirotor UAV has the ability to generate thrust in *any* direction by changing it's roll- and pitch-angle. Consider the following

$$\dot{\mathbf{p}} = \mathbf{v} \tag{5.13}$$

$$\dot{\Theta} = \mathbf{T}(\Theta)\boldsymbol{\omega} \tag{5.14}$$

where $\mathbf{p} = \mathbf{p}_b^n = \begin{bmatrix} x^n & y^n & z^n \end{bmatrix} \in \mathbb{R}^3$ is the position in NED.

As in [33], the multirotor dynamics can be written in the following form

$$\dot{\mathbf{p}} = \mathbf{v} \tag{5.15}$$

$$m_{uav}\dot{\mathbf{v}} = m_{uav}\mathbf{g} + \mathbf{R}_b^n(\Theta)\mathbf{f} \tag{5.16}$$

$$\dot{\Theta} = \mathbf{T}(\Theta)\boldsymbol{\omega} \tag{5.17}$$

$$I\dot{\boldsymbol{\omega}} = \mathbf{S}(I\boldsymbol{\omega})\boldsymbol{\omega} + \mathbf{M} \tag{5.18}$$

Where $\mathbf{f} = \begin{bmatrix} 0 & 0 & -f \end{bmatrix}^\top$ and $\mathbf{M} = \begin{bmatrix} M_x & M_y & M_z \end{bmatrix}^\top$ are the force vector and moments created by the motors respectively. The conversion between motor forces and equivalent forces and moments in $\{b\}$ is given by

$$\begin{bmatrix} f \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ L_s & -L_s & -L_s & L_s \\ L_f & L_f & -L_b & L_b \\ -c & c & -c & c \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{5.19}$$

The force allocation matrix is specific for this particular configuration of rotors, and the constants are defined in table 5.1.

Symbol	Description
$f_i, \quad i \in \{1,2,3,4\}$	Magnitude of motor thrusts (enumerated clockwise from front left motor)
L_f	Lever arm front
L_b	Lever arm back
L_s	Lever arm side

Table 5.1: A definition of the symbols used in the 3D model

5.2 Contact Force Augmentation

This section is inspired by [33]; however, it differs with respect to the coordinate systems used and model of compliant behavior.

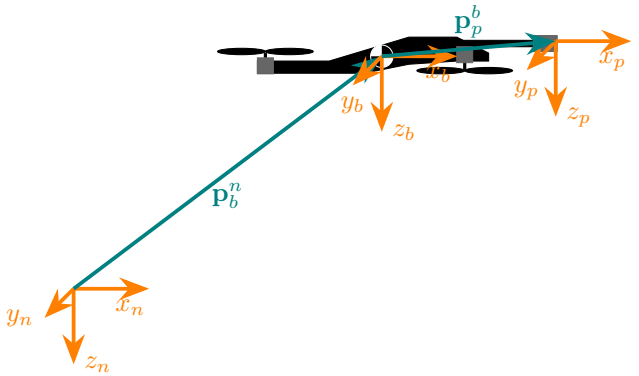


Figure 5.2: 3D model for contact

In this section, eq. (4.6) is augmented with external forces $\mathbf{f}_e$ and moments $\mathbf{M}_e$, such that

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ m_{uav}\dot{\mathbf{v}} &= m_{uav}\mathbf{g} + \mathbf{R}_b^n(\Theta)\mathbf{f} + \mathbf{f}_e \\ \dot{\Theta} &= \mathbf{T}(\Theta)\boldsymbol{\omega} \\ I\dot{\boldsymbol{\omega}} &= \mathbf{S}(I\boldsymbol{\omega})\boldsymbol{\omega} + \mathbf{M} + \mathbf{M}_e \end{aligned} \tag{5.20}$$

According to the coordinate frames in fig. 5.2, these forces and moments can be found as [33]

$$\mathbf{f}_e = \mathbf{R}_b^i \mathbf{R}_p^b \mathbf{f}_c^p \tag{5.21}$$

$$\mathbf{M}_e = \mathbf{R}_b^i \mathbf{R}_p^b \mathbf{M}_c^p + \mathbf{R}_b^i \left(\mathbf{R}_p^b \mathbf{f}_c^p \times \mathbf{p}_p^b \right) \tag{5.22}$$

This includes the entire wrench applied from the environment to the drone. However, the moments $\mathbf{M}_c$ that are directly applied by the environment are very small for the interactions considered in this thesis, and are omitted in the following. The forces and moments created by the gravitational force of the probe are also assumed negligible.

5.2.1 Compliant Behavior

The contact force is given by $\mathbf{f}_c^p = \mathbf{R}_{ee}^p \begin{bmatrix} -f_c & 0 & 0 \end{bmatrix}^\top$, where $\mathbf{R}_{ee}^p$ is the rotation matrix between two frames with origin in the end-effector. The first frame $\{p\}$ is shown in fig. 5.2, and is the frame with x-axis aligned with the vector from *the origin* to the end-effector tip. The second frame $\{ee\}$ is the frame with x-axis aligned with the vector from *the probe attachment point* and the end-effector tip. The magnitude f_c is found the same way as in previous chapter, and is given by equation eq. (4.13)

As with the 2D model, it is a possibility to model the compliant behavior of the drone as a separate subsystem to include more advanced compliant subsystems as in [33].

5.3 Simulation Validation

Similar to the previous chapter, the model is implemented as a simulator in MATLAB/Simulink. To get the simulation as accurate as possible, realistic motor simulations have been included. The input to the UAV model is the individual motor voltages. The model is implemented according to the common motor model [12]

$$\dot{I}_m = \frac{1}{L_m}(-R_m I_{m,i} + V_m - K_e \Omega_{m,i}) \tag{5.23}$$

$$\dot{\Omega}_m = \frac{1}{J_m}(-B_m \Omega_{m,i} + K_t I_{m,i} - K_q \Omega_{m,i}^2) \tag{5.24}$$

Where $I_{m,i}$ and V_m is the individual motor currents and voltages and $\Omega_{m,i}$ are the angular velocities of the rotors. The motor thrusts are then given by

$$f_i = K_T \Omega_{m,i}^2 \tag{5.25}$$

The values of the constants used in the simulations are adapted from [6] and given in table 5.2.

Symbol	Description	Value
L_m	Electrical inductance	$8.4 \times 10^{-4}\,\mathrm{H}$
R_m	Electrical resistance	$0.168\,\Omega$
K_e	EFM constant	$1.248 \times 10^{-2}\,\mathrm{V\,s\,rad^{-1}}$
J_m	Inertia seen by the motor	$4.2 \times 10^{-7}\,\mathrm{kg\,m^2}$
B_m	Viscous damping ratio	$4.2 \times 10^{-5}\,\mathrm{N\,m\,s}$
K_t	Motor torque constant	$1.0608 \times 10^{-2}\,\mathrm{N\,m\,A^{-1}}$
K_q	Motor thrust constant	$9.2041 \times 10^{-8}\,\mathrm{kg\,m^2\,rad^{-2}}$
K_T	Thrust constant	$2.7517 \times 10^{-5}\,\mathrm{kg\,m\,rad^{-1}}$

Table 5.2: Numerical values of constants in motor and propeller models

A low-level attitude controller was also implemented according to [17].

In the following simulation, the pitch reference is set to $-5°$ for 1 second starting at $t = 11.5\,\mathrm{s}$ and the yaw reference ψ was set constant at $5°$. This causes the drone to collide with the inspection surface at an angle. The placement of the inspection surface can be seen in fig. 5.4.

As with the 2D simulations, fig. 5.3 the interaction pushes the drone backwards, and causes a rotation of the pitch angle θ. The velocity of the drone in combination with the angled approach causes the drone to rotate towards the inspection surface. These effects can also clearly be seen in fig. 5.4, which depicts the drone before, during and after the interaction. The roll angle is also slightly perpetuated as a result of the interaction. Afterwards the attitude controller corrects the attitude and the drone drifts away from the wall. This behavior fits well with the physical interpretation and validates the performance of the model.

Friction was not included in the model; however, the contact forces are modeled to work along the probe, instead of normal to the surface. This is slightly non-physical, but the resulting behavior is similar to having friction. Modeling and simulating the friction correct is very difficult. Hence, this has not been the focus in this part, as the model is mainly used for control synthesis and

initial validation. As mentioned, a more thorough evaluation of the controller performance will be conducted in part Appendix Part I, where correct friction simulations are included.

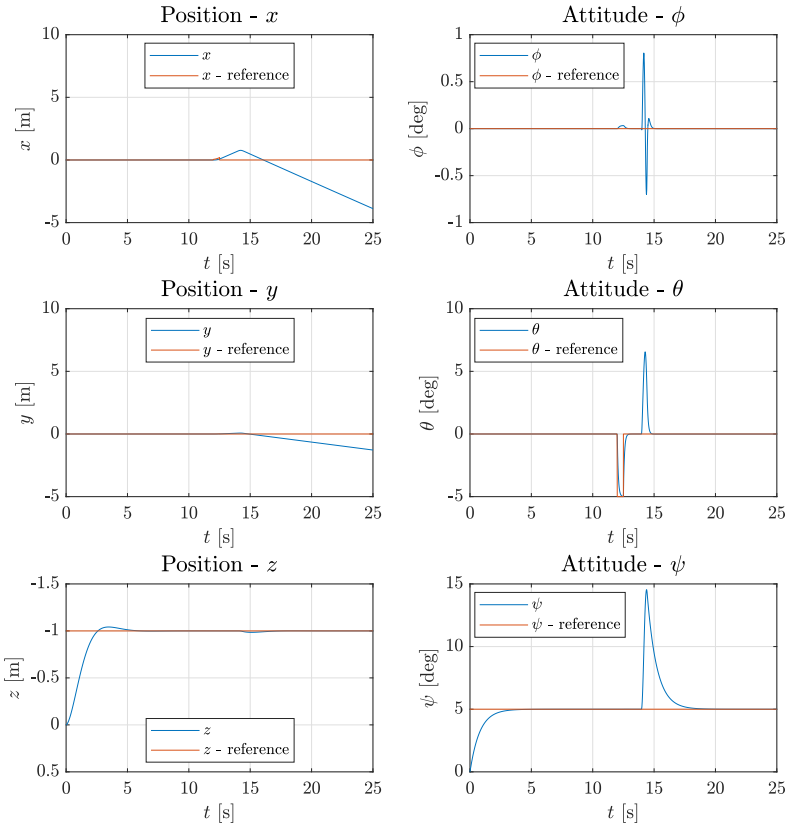
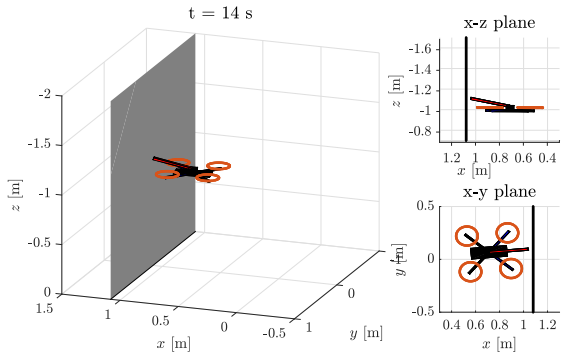
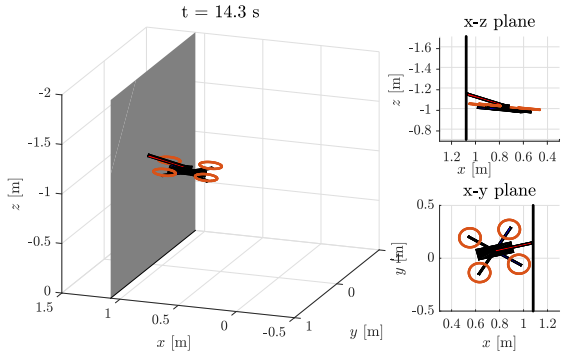


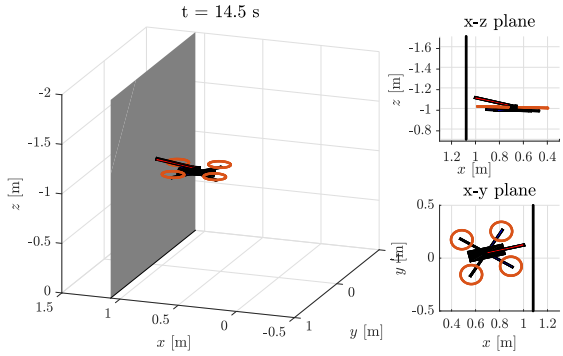
Figure 5.3: State variables during simulation



(a) Time step $t = 14.0\,\mathrm{s}$



(b) Time step $t = 14.3\,\mathrm{s}$



(c) Time step $t = 14.5\,\mathrm{s}$

Figure 5.4: Momentary snapshots of the UAV before, during and after interaction

Chapter 6

Udwadia Kalaba

In 1992, Udwadia and Kalaba [69] proposed a way of dealing with constraints on dependent generalized coordinates. This method uses the equations for the constrained acceleration to directly compute the constraint forces. The framework is flexible, and the transition from constrained interaction to free flight can be handled by removing the constraint from the equation when the velocity is no longer in the direction of the constraining object. The most prominent downside to this method, is that analytical expressions are usually more complex than for independent coordinates, and therefore less suitable for control purposes.

The motivation for using the Udwadia Kalaba equations, is that they will provide an accurate method for simulating the constrained motion during interaction. An example of where this could be relevant, is detailed studies of how attempts at correcting positional errors in the direction parallel to the inspection surface influences the stability of the interaction. However, the following will only present the derivation of the necessary equations, and simulation implementation is left as future work.

This chapter will first present the necessary background for the Udwadia Kalaba equations, before giving a simple example showcasing the main differences to more standard methods. Thereafter, the Udwadia Kalaba constraint equations are derived for a UAV interacting in constrained motion with an inspection surface and finally a short discussing is presented.

6.1 Background

In this section the necessary background for the Udwadia Kalaba equations is presented along with an explanatory example.

6.1.1 Principle

This section follows the derivation in [43]. Let the unconstrained system be defined as

$$\mathbf{M}\ddot{\mathbf{q}}_u = \mathbf{Q} \tag{6.1}$$

where $\mathbf{q}_u \in \mathbb{R}^n$ is the unconstrained generalized coordinates, and $\mathbf{Q} \in \mathbb{R}^n$ are generalized forces. The system has p constraints, such that it can be written on the form

$$\mathbf{A}(\mathbf{q}, \dot{\mathbf{q}}, t)\ddot{\mathbf{q}} = \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}, t) \tag{6.2}$$

where $\mathbf{A} \in \mathbb{R}^{p\times n}$ and $\mathbf{b} \in \mathbb{R}^p$ are matrices and $\mathbf{q} \in \mathbb{R}^n$ are the generalized coordinates of the constrained motion of eq. (6.1).

The system in eq. (6.1) can be converted to a constrained system by adding constraint forces $\mathbf{Q}_c \in \mathbb{R}^n$, such that

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{Q} + \mathbf{Q}_c \tag{6.3}$$

In [25], eq. (6.3) is solved by applying Gauss's principle of Least Constraints. This principle states that the acceleration of the constrained system follows the vector closest to the unconstrained acceleration, that satisfies the constraints. The minimization problem that follows from this can be solved using the Moore-Penrose pseudoinverse. Moreover, the constrained system's acceleration $\ddot{\mathbf{q}}$ can be found from (see [25]):

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_u + \mathbf{M}^{-1/2}\left(\mathbf{A}\mathbf{M}^{-1/2}\right)^{+}(\mathbf{b} - \mathbf{A}\ddot{\mathbf{q}}_u) \tag{6.4}$$

where $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. From this the constraint force $\mathbf{Q}_c$ can be found as

$$\mathbf{Q}_c = \mathbf{M}^{1/2}\left(\mathbf{A}\mathbf{M}^{-1/2}\right)^{+}(\mathbf{b} - \mathbf{A}\ddot{\mathbf{q}}_u) \tag{6.5}$$

For a complete derivation with proofs see [69].

6.1.2 A Simple Example

This section is adapted from the example given in [43], and the purpose is to give the reader insight into some of the key differences of the Udwadia Kalaba approach to other, more used methods.

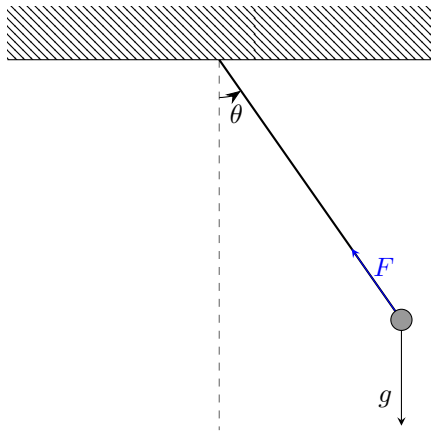


Figure 6.1: Pendulum system

Consider the classical pendulum system illustrated in fig. 6.1, where a pendulum of mass m is attached to a point. In order to model the system, a set of coordinates must be chosen to describe the configuration. The classical choice is to use the angle θ, which uniquely describes the position of the pendulum. Then, by defining $q = \theta$, q becomes the independent coordinates of this 1 DOF system.

However, another choice is to describe the system using (x, y), i.e. the position of the point-mass in cartesian coordinates. From many perspectives this is the intuitive way of describing the system, as the mass inherently has a position in cartesian space that might be of great interest. However, the point-mass cannot move freely in the cartesian space and not all configurations (x, y) are reachable. More specifically, it is restricted to the circle $x^2 + y^2 = L_s^2$. If $q = [x, y]^\top$ is chosen as the generalized coordinates, the system is described with 2 coordinates. However, it is important to note that our system still only has 1 DOF, and thus 1 constraint is also present.

This type of generalized coordinates are called constrained generalized coordinates, or dependent generalized coordinates.

The following shows the derivation of equations of motion for a different choice of generalized coordinates and methods of derivation.

Lagrange With Independent Coordinates

The Lagrangian of the system can be defined as

$$L = \underbrace{\frac{1}{2} m L_s^2 \dot{q}^2}_{\text{Kinetic}} \underbrace{-mgL_s(1-\cos q)}_{\text{Potential}} \tag{6.6}$$

with $q = \theta$. Using Euler-Lagrange Equations, $\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_j}\right) = \frac{\partial L}{\partial q_j}$, the equations of motions are found as

$$\ddot{\theta} + \frac{g}{L_s}\sin\theta = 0 \tag{6.7}$$

These equations are simple; however, they do not provide immediate insight into the (x, y) placement of the point-mass. For more complicated systems the relation between the independent coordinates and the position in the cartesian space become even more convoluted.

Lagrange With Cartesian Coordinates

When using Lagrange mechanics, the constraints are usually written on the form

$$\sum_{j=1}^{N} a_{ij}(\mathbf{q}, t)\dot{q}_j + b_i(\mathbf{q}, t) = 0 \tag{6.8}$$

The constraint is given as

$$x^2 + y^2 = L_s^2 \tag{6.9}$$

However, differentiating once with respect to time yields

$$x\dot{x} + y\dot{y} = 0 \tag{6.10}$$

And by inspection it is seen that $a_{11} = x$ and $a_{22} = x$ and all other functions a and b are zero. The Lagrangian can in this case be written as.

$$L = \underbrace{\frac{1}{2}m(\dot{x}^2 + \dot{y}^2)}_{\text{Kinetic}} - \underbrace{mg(L_s - y)}_{\text{Potential}} \tag{6.11}$$

And once again using Euler-Lagrange equations, $\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_j}\right) = \frac{\partial L}{\partial q_j}$, the equations of motion are found as

$$m\ddot{x} = \lambda a_x = \lambda x \tag{6.12}$$
$$m\ddot{y} = mg + \lambda a_y = mg + \lambda y \tag{6.13}$$

where λ is the constraint factor. This can be written in a more convenient vector form as

$$\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ mg \end{bmatrix} - \lambda \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.14}$$

The tricky part about this method is to find λ, which in general is non-trivial.

Using Udwadia Kalaba's Equations

Also here, $\mathbf{q}$ is chosen as $\mathbf{q} = \begin{bmatrix} x & y \end{bmatrix}^\top$ under the constraint $x^2 + y^2 = L_s^2$. According to eq. (6.2), the second derivatives of the constraints are need. Differentiating the constraint with respect to time yields

$$2x\dot{x} + 2y\dot{y} = 0 \tag{6.15}$$

And once more

$$2\dot{x}^2 + 2x\ddot{x} + 2\dot{y}^2 + 2y\ddot{y} = 0 \tag{6.16}$$
$$\underbrace{\begin{bmatrix} x & y \end{bmatrix}}_{:=A} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \underbrace{-\dot{x}^2 - \dot{y}^2}_{:=b} \tag{6.17}$$

The equations of the unconstrained system is

$$\underbrace{\begin{bmatrix} m & 0 \\ 0 & m \end{bmatrix}}_{:=\mathbf{M}} \mathbf{a} = \begin{bmatrix} 0 \\ mg \end{bmatrix} \tag{6.18}$$

as gravity is the only force working.

Inserting into the Udwadia Kalaba equations yields

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ g \end{bmatrix} + \mathbf{M}^{-1/2} \left(\mathbf{A}\mathbf{M}^{-1/2} \right)^{+} \left(b - \mathbf{A} \begin{bmatrix} 0 \\ g \end{bmatrix} \right) \tag{6.19}$$

Evaluating the Moore-Penrose psudoinverse gives

$$\left(\mathbf{A}\mathbf{M}^{-1/2} \right)^{+} = \frac{1}{m^{-1}x^2 + m^{-1}y^2} \begin{bmatrix} m^{-1/2}x \\ m^{-1/2}y \end{bmatrix} \tag{6.20}$$

inserting and rearranging the equations yield

$$\mathbf{M} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ mg \end{bmatrix} - m \frac{\dot{x}^2 + \dot{y}^2 + gy}{L_s^2} \begin{bmatrix} x \\ y \end{bmatrix} \tag{6.21}$$

This is similar as the previous section; however, λ is now given explicitly as

$$\lambda = -m \frac{\dot{x}^2 + \dot{y}^2 + gy}{L_s^2} \tag{6.22}$$

6.2 Derivation

In this section the Udwadia Kalaba equations will be derived for UAV contact with the environment.

The general coordinates are defined as

$$\mathbf{q} = \begin{bmatrix} \mathbf{p}_b^n & \mathbf{\Theta} \end{bmatrix}^{\top} \tag{6.23}$$

And the model of the unconstrained motion of the drone is given as in section 5.1.

$$\mathbf{M}_b \dot{\nu}_b + \mathbf{C}\nu = \boldsymbol{\tau}_{RB} \tag{6.24}$$

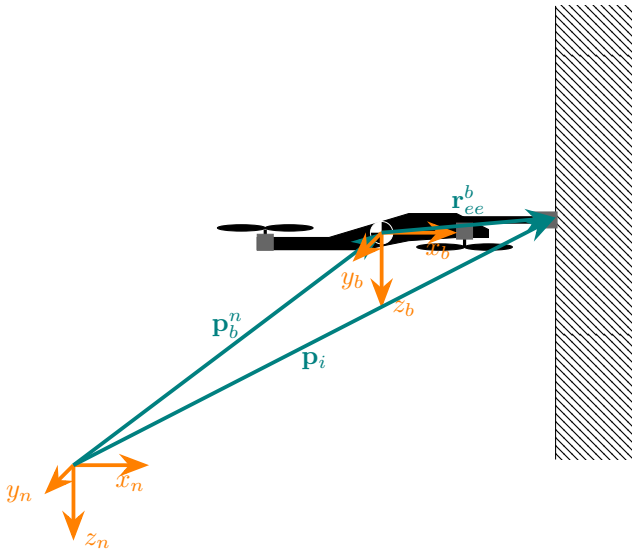


Figure 6.2: Frames and vectors used in the Udwadia Kalaba equations

However, $\dot{\nu}_b$ is differentiated in body-frame, hence it cannot be used as generalized acceleration. The solution to this is given in [43], by using that the equations of motion can be written on the form

$$\mathbf{M}_b \begin{bmatrix} \mathbf{a}_b^b \\ \boldsymbol{\alpha}_b^b \end{bmatrix} = \boldsymbol{\tau}_{RB}$$

where $\mathbf{a}_b^b$ is the acceleration represented in the body-fixed frame (specific force), and $\boldsymbol{\alpha}_b^b$ is specific torque. This will be used as generalized acceleration in the derivation in the following section.

6.2.1 Constraint Equation Derivation

Referring to fig. 6.2, the point-of-impact $\mathbf{p}_i$ is the vector between the end-effector and the point-of-impact can be written as

$$\mathbf{L}^n = \mathbf{p}_b^n + \mathbf{r}_{ee}^n - \mathbf{p}_i \tag{6.25}$$

Inserting the rotation matrix this gives

$$\mathbf{L}^n = \mathbf{p}_b^n + \mathbf{R}_b^n \mathbf{r}_{ee}^b - \mathbf{p}_I \tag{6.26}$$

Constrain the end effector at the point of impact yields

$$0 = g = \|\mathbf{L}^n\|^2 = (\mathbf{L}^n)^\top \mathbf{L}^n \tag{6.27}$$

Differentiating g w.r.t. time gives

$$\dot{g} = \frac{{}^n\mathrm{d}}{\mathrm{d}t} g = 2\dot{\mathbf{L}}^\top \mathbf{L} = 0 \tag{6.28}$$

And one more differentiation yields

$$\ddot{g} = 2\ddot{\mathbf{L}}^\top \mathbf{L} + 2\dot{\mathbf{L}}^\top \dot{\mathbf{L}} = 0 \tag{6.29}$$

Then the derivative of $\mathbf{L}$ is found as

$$\dot{\mathbf{L}} = \dot{\mathbf{p}}_b^n + \dot{\mathbf{R}}_b^n \mathbf{r}_{ee}^b \tag{6.30}$$

where it has been used that $\dot{\mathbf{p}}_i = 0$, since the point of impact is stationary. Differentiating $\mathbf{L}$ once more then gives

$$\ddot{\mathbf{L}} = \ddot{\mathbf{p}}_b^n + \ddot{\mathbf{R}}_b^n \mathbf{r}_{ee}^b \tag{6.31}$$

The first and second derivative of $\mathbf{R}_b^n \mathbf{r}_{ee}^b$ can be found as

$$\dot{\mathbf{R}}_b^n \mathbf{r}_{ee}^b = \frac{{}^n\mathrm{d}}{\mathrm{d}t} \mathbf{R}_b^n \mathbf{r}_{ee}^b = \mathbf{R}_b^n \mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right) \mathbf{r}_{ee}^b = -\mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_{ee}^b\right) \boldsymbol{\omega}_{b/n}^b \tag{6.32}$$

$$\ddot{\mathbf{R}}_b^n \mathbf{r}_{ee}^b = -\dot{\mathbf{R}}_b^n \mathbf{S}\left(\mathbf{r}_{ee}^b\right) \boldsymbol{\omega}_{b/n}^b - \mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_{ee}^b\right) \frac{{}^n\mathrm{d}}{\mathrm{d}t} \boldsymbol{\omega}_{b/n}^b \tag{6.33}$$

Then it can be utilized that $\frac{{}^n\mathrm{d}}{\mathrm{d}t}\boldsymbol{\omega}_{b/n}^b = \frac{{}^b\mathrm{d}}{\mathrm{d}t}\boldsymbol{\omega}_{b/n}^b$, such that

$$\ddot{\mathbf{R}}_b^n \mathbf{r}_{ee}^b = \mathbf{R}_b^n \mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right) \mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right) \mathbf{r}_{ee}^b - \mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_{ee}^b\right) \dot{\boldsymbol{\omega}}_{b/n}^b \tag{6.34}$$

This makes $\mathbf{L}$ a function of $\ddot{\mathbf{p}}_b^n$ and $\dot{\boldsymbol{\omega}}_{b/n}^b$. This needs to be converted to generalized coordinates, represented in $\{b\}$. Following [43], this can be done by

$$\mathbf{a}_b^b = \mathbf{R}_b^n \ddot{\mathbf{p}}_b^n \tag{6.35}$$

Then, by using $\mathbf{I}_b\boldsymbol{\alpha}_b^b = \mathbf{m}_{rb}^b$, where $\mathbf{m}_{rb}^b$ are the moments in the body fixed frame, the equations become

$$\mathbf{I}_b\dot{\boldsymbol{\omega}}_{b/n}^b = \mathbf{m}_{rb}^b - \mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{I}_b\boldsymbol{\omega}_{b/n}^b \tag{6.36}$$

$$\dot{\boldsymbol{\omega}}_{b/n}^b = \underbrace{\mathbf{I}_b^{-1}\mathbf{m}_{rb}^b}_{:=\boldsymbol{\alpha}_b^b} - \mathbf{I}_b^{-1}\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{I}_b\boldsymbol{\omega}_{b/n}^b \tag{6.37}$$

Inserting eqs. (6.35) and (6.37) into eq. (6.34) yields

$$\begin{aligned}\ddot{\mathbf{R}}_b^n\mathbf{r}_{ee}^b = &\mathbf{R}_b^n\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{r}_{ee}^b \\ &- \mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\boldsymbol{\alpha}_b^b \\ &- \mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\mathbf{I}_b^{-1}\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{I}_b\boldsymbol{\omega}_{b/n}^b\end{aligned} \tag{6.38}$$

Further, inserting this into eq. (6.29) reveals that

$$\begin{aligned}\ddot{g} &= 2\dot{\mathbf{L}}^\top\dot{\mathbf{L}} + 2\ddot{\mathbf{L}}^\top\mathbf{L} \\ &= 2\dot{\mathbf{L}}^\top\dot{\mathbf{L}} + 2\mathbf{L}^\top\ddot{\mathbf{L}} \\ &= 2\dot{\mathbf{L}}^\top\dot{\mathbf{L}} + 2\mathbf{L}^\top\left(\ddot{\mathbf{p}}_b^n + \ddot{\mathbf{R}}_b^n\mathbf{r}_{ee}^b\right) \\ &= 2\dot{\mathbf{L}}^\top\dot{\mathbf{L}} + 2\mathbf{L}^\top\left(\ddot{\mathbf{p}}_b^n + \mathbf{R}_b^n\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{r}_{ee}^b\right. \\ &\quad \left.-\mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\boldsymbol{\alpha}_b^b - \mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\mathbf{I}_b^{-1}\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{I}_b\boldsymbol{\omega}_{b/n}^b\right) \\ &= 2\dot{\mathbf{L}}^\top\dot{\mathbf{L}} \\ &\quad + 2\mathbf{L}^\top\left(\ddot{\mathbf{p}}_b^n - \mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\boldsymbol{\alpha}_b^b\right) \\ &\quad + 2\mathbf{L}^\top\Upsilon_b \\ &= 2\dot{\mathbf{L}}^\top\dot{\mathbf{L}} + 2\mathbf{L}^\top\left(\mathbf{R}_b^n\mathbf{a}_b^b - \mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\boldsymbol{\alpha}_b^b\right) + 2\mathbf{L}^\top\Upsilon_b\end{aligned}$$

where:

$$\Upsilon_b = \mathbf{R}_b^n\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{r}_{ee}^b - \mathbf{R}_b^n\mathbf{S}\left(\mathbf{r}_{ee}^b\right)\mathbf{I}_b^{-1}\mathbf{S}\left(\boldsymbol{\omega}_{b/n}^b\right)\mathbf{I}_b\boldsymbol{\omega}_{b/n}^b \tag{6.39}$$

This can be factored as

$$\mathbf{A}\ddot{\mathbf{q}} = \mathbf{b} \tag{6.40}$$

With $\mathbf{A}$ and $\mathbf{b}$ as

$$\mathbf{A} = 2\mathbf{L}^\top \begin{bmatrix} \mathbf{R}_b^n & -\mathbf{R}_b^n \mathbf{S}\left(\mathbf{r}_{ee}^b\right) \end{bmatrix} \tag{6.41}$$

$$\mathbf{b} = 2\dot{\mathbf{L}}^\top \dot{\mathbf{L}} + 2\mathbf{L}^\top \Upsilon_b \tag{6.42}$$

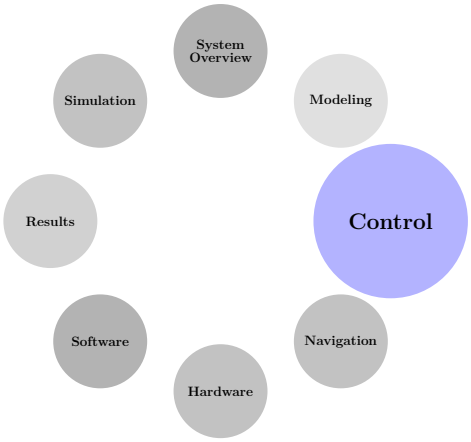
This concludes the derivation of the Udwadia Kalaba equations for the constrained motion of the drone.

6.3 Discussion and Future Work

As mentioned previously, the models based on the Udwadia Kalaba is not well suited for control synthesis. In [43], the algebraic complexity in the resulting dynamic equations are shown. As mentioned in the introduction of this chapter, the motivation behind these derivation is that this provide an accurate method for simulating the constrained motion during interaction. This has not yet been implemented for simulations, but the derivations made in this thesis lays a solid foundation for future work on the topic.

Part III

Control



Chapter 7

Background Theory

This chapter contains a very brief introduction to force control and hereunder impedance control relevant to this thesis. For a more thorough introduction, the reader is referred to [70] and [39], which the following sections are based on.

7.1 Force Control

Force control is an essential part of robots if they are to accomplish tasks at the same or higher level as humans. In everything from grasping an egg, via polishing a car, to pushing a cart, the concept of force is essential for a safe and successful completion of the mission. Too much force, and the egg breaks; too little force, and the egg falls out of the robotic arm. Force control differs from motion control with regards to the sensitivity to errors in the desired trajectory. If a perfect model of the environment and the controlled system can be obtained, a trajectory can be planned and executed successfully without any errors. However, assume now that a screw is to be tightened using a positional approach. The relative positioning of the parts needs to have an accuracy significantly larger than the tolerance of the mechanical parts. Given the exact location of the screw, the manipulator needs to guide all the parts (its own links/joints and the screw) with the same level of accuracy. However, in practice small errors may give rise to contact forces and moments, which causes the end effector to deviate from the desired trajectory. The control system reacts to reduce such deviations, which causes a build-up of the contact force until something breaks. The higher the stiffness of the environment or the end effector the more prominent these challenges will be.

To better understand the different scenarios, [70] classifies four different types of interactions with the environment:

Constrained motion:
Corresponds to the situation where the robot is in contact with a stiff surface. This will constrain the paths that the end effector can follow, also know as kinematic constraints.

Inertial:
Dynamic interaction between the robot and the environment, for example pushing a box.

Dissipative:
Also dynamic interaction between the robot and the environment, but here the robot experiences friction when it is sliding along the surface.

Elastic:
Dynamic interaction between the robot and the environment, where the environment is elastically compliant.

The four types are characterized by the need for different types of control. The scenario studied in this thesis is clearly in the first category, as the environment does not exhibit any dynamic behavior. Admittance control and impedance control are examples of control types applicable for this category. These will be discussed further; however, first a distinction between passive and active interaction control is made.

7.1.1 Interaction Control

In interaction control, a clear distinction is drawn between passive and active interaction control, mainly on the basis of the sensors available ([70]):

Passive interaction control
The trajectory of the end effector is modified by the compliant behavior when the robot interacts with the environment. This compliant behavior might be due to the structural compliance and/or to the compliance of the controllers. The simplicity of this method is its largest strength, as it is cheaper and easy to make and also does not require any additional sensors. However, This method does lack flexibility, since for every task a specific robot has to be designed. Since no forces/torques are measured, it also cannot guarantee that high contact forces will not occur.

Active interaction control
Requires a feedback-controller with force measurements to modify desired trajectories to achieve compliant behavior of the end effector. This helps overcome the disadvantages with passive interaction control, but is slower, more expensive and more complex. It is essential that this method is used in combination with passive compliance to achieve better task execution speed and disturbance rejection capability at a hardware level.

It should be noted that in this context the word passive is not a reference to the general area of passivity based control [42], but rather to the fact that the interaction forces are not directly measured and compensated for.

7.2 Impedance Control

Both force control in general and impedance especially is concerned with handling interaction between systems, and therefore it is essential to understand how the relations between subsystems can be expressed.

One of the ways these interactions can be described, is using the concept of flow and effort variables. The effort can for example be a force for a translational system or a voltage for an electrical system, while the flow is a velocity or a current respectively. The main idea is that for each degree of freedom in the interaction between two or more physical systems, the effort has to be equal, while the flows has to sum to zero. This approach to modeling is not uncommon and can among others be used for mechanical, electrical, thermal and fluid systems. In fact, many advanced simulation engines uses this to facilitate a modular and flexible system of components. One example of this is the Modelica[1] programming language used in the simulation software Dymola[2].

Several important constraints on the behavior of physical systems can be identified using this approach. The instantaneous power flow between systems (e.g., a physical system and its environment) is always definable as the product of the two conjugate variables, effort and flow. Another very important physical constraint is that no *one* system may determine *both* variables. For example, for a general translational system along any degree of freedom, the system may impose a force on its environment *or* impose a displacement or velocity on it, but *not* both. Seen from the environment, physical systems can only be one of two types ([70]):

[1] https://www.modelica.org/
[2] https://www.3ds.com/products-services/catia/products/dymola/

Admittances: Accept effort (e.g. force) as inputs and yields corresponding flow (e.g. motion) as outputs.

Impedances: Accept flow (e.g., motion) as inputs and yield effort (e.g., force) as outputs.

When considering typical motion control, the system is modeled as an admittance. The system yields its own flow variable while accepting effort as input. However, as discussed above, this is not optimal when the interaction forces are of importance. The interaction is executed without considerations for the interactions forces. In contrast, impedance control is better suited for these situations, because during interaction it will accept the motion constraints given by other systems and rather yield the force. Impedance is a well known concept in the area of robotic manipulations, and the following presents a brief example of impedance control for a classic manipulator. Consider the state space formulation of a robotic arm ([70])

$$\mathbf{\Lambda}(q)\dot{\mathbf{v}}_e + \mathbf{\Gamma}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{v}_e + \eta(\mathbf{q}) = \mathbf{h}_c - \mathbf{h}_e \tag{7.1}$$

where $\mathbf{v}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$ is the velocity of the end effector, the matrix $\mathbf{J}$ is the $6 \times n$ end effector geometric Jacobian. Also, $\mathbf{\Lambda}(\mathbf{q}) = \left(\mathbf{J}\mathbf{H}(\mathbf{q})^{-1}\mathbf{J}^\top\right)^{-1}$ is the 6×6 operational space matrix, $\mathbf{\Lambda}(\mathbf{q}, \mathbf{q}) = \mathbf{J}^{-\top}\mathbf{C}(\mathbf{q}, \mathbf{q})\mathbf{J}^{-1} - \mathbf{\Lambda}(\mathbf{q})\dot{\mathbf{J}}\mathbf{J}^{-1}$ is the wrench including centrifugal and Coriolis forces, $\boldsymbol{\eta}(\mathbf{q}) = \mathbf{J}^{-\top}\mathbf{g}(\mathbf{q})$ is the wrench of gravitational effects. $\mathbf{H}(\mathbf{q})$, $\mathbf{C}(\mathbf{q}, \mathbf{q})$ and $\mathbf{g}(\mathbf{q})$ are the corresponding quantities defined in the joint-space. The control law is defined as ([70])

$$\mathbf{h}_c = \mathbf{\Lambda}(\mathbf{q})\boldsymbol{\alpha} + \Gamma(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{h}_e \tag{7.2}$$

where $\mathbf{h}_e$ is a new virtual input. The details of the derivations from [70] are skipped here; however, by choosing $\mathbf{h}_e$ carefully this results in a state-space formulation in the end effector frame as

$$\mathbf{K_M}\Delta\dot{\mathbf{v}}^e_{de} + \mathbf{K}_D\Delta v^e_{de} + \mathbf{h}^e_\Delta(\Delta\mathbf{p}^e_{de}) = \mathbf{h}^e_e \tag{7.3}$$

Here the equivalence to an 6-DOF impedance is clear. In the case of a robotic arm, it is possible to dissipate forces through the base of the arm; however, for an underactuated system such as an UAV this is not necessarily possible. For a drone, this means that the coupling between forces and attitude has to be handled. One of the benefits of the impedance controller is that the impedance control is a passive force control technique, allowing for a simpler hardware design as no force/torque-measurements are needed.

Chapter 8

Controller 2D

This chapter will present an impedance based controller for the 2D model, and as mentioned in section 4.2, the following is inspired by [32]. After that, the simulation results will be presented along with a short discussion.

8.1 Controller Derivation

By first assuming that an attitude controller is in place, such that $\theta = \theta^*$, the model can be considered to have virtual inputs $u = \begin{bmatrix} \theta & f^b \end{bmatrix}^T$. Based on eq. (4.4), the dynamics can then be written as

$$M_{uav}\ddot{x} = -f^b \sin\theta - f_c \cos(\theta + \theta_p) \tag{8.1}$$

$$M_{uav}\ddot{z} = -f^b \cos\theta + M_{uav}g + f_c \sin(\theta + \theta_p) \tag{8.2}$$

For each degree of freedom ζ, this can be generalized as [32]

$$M_\zeta \ddot{\zeta} = u_\zeta + d_\zeta \tag{8.3}$$

for $\zeta \in \{x, z\}$, where

$$M_\zeta = M_{uav} \tag{8.4}$$
$$u_x = -f^b \sin\theta \tag{8.5}$$
$$u_z = -f^b \cos\theta + M_{uav} g \tag{8.6}$$
$$d_x = f_c \cos(\theta + \theta_p) \tag{8.7}$$
$$d_z = f_c \sin(\theta + \theta_p) \tag{8.8}$$

Then, let the control law be given as

$$u_\zeta = -k_p^\zeta(\zeta - \zeta^*) - k_d^\zeta \dot{\zeta} \tag{8.9}$$

Where ζ^* is the desired placement. For each degree of freedom, this yields the following dynamics:

$$M_\zeta \ddot{\zeta} + k_d^\zeta \dot{\zeta} + k_p^\zeta(\zeta - \zeta^*) = d_\zeta \tag{8.10}$$

And the impedance equivalence becomes apparent. This equivalence is illustrated in fig. 8.1, where it is shown how the drone reacts to a disturbance from the environment when controlled as an impedance. By tuning the parameters in the controller, the stiffness and damping of the equivalent impedances are adjusted. This will in turn determine the resulting reaction to the disturbance d_ζ from the environment.

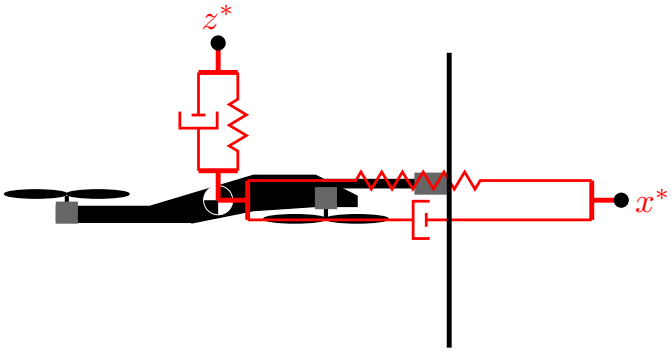


Figure 8.1: Impedance model of drone

The virtual inputs can then be found solving eqs. (8.5) and (8.6) for θ and f^b

$$\theta = \text{atan2}\left(u_x, u_z - M_{uav}g\right) \tag{8.11}$$

$$f^b = \sqrt{u_x^2 + (u_z - M_{uav}g)^2} \tag{8.12}$$

This is valid for all $u_x, u_z \in \mathbb{R}$, except when $u_z = M_{uav}g$. This corresponds to a free-fall scenario. This is easily avoided, since this is far from standard operating conditions. However, care should be taken when implementing and tuning the controller to avoid numerical instabilities.

Passivity Proof

The controller above can also be interpreted as a passivity based controller, and the following will present a proof of passivity.

The storage function is defined as [32]

$$V = \mathcal{K}(\dot{\zeta}) + \mathcal{P}(\zeta) \tag{8.13}$$

where $\mathcal{K}(\dot{\zeta})$ is the kinetic energy and $\mathcal{P}(\zeta)$ is the potential energy defined as

$$\mathcal{K}(\dot{\zeta}) = \frac{1}{2}M_\zeta \dot{\zeta}^2 \tag{8.14}$$

$$\mathcal{P}(\zeta) = \frac{1}{2}k_p^\zeta(\zeta - \zeta^*)^2 \tag{8.15}$$

Such that $V \geq 0$.

Differentiating along the trajectory yields

$$\begin{aligned}
\dot{V} &= M_\zeta \dot{\zeta}\ddot{\zeta} + k_p^\zeta(\zeta - \zeta^*)\dot{\zeta} && (8.16)\\
&= \dot{\zeta}(u_\zeta + d_\zeta) + k_p^\zeta(\zeta - \zeta^*)\dot{\zeta} && (8.17)\\
&= \dot{\zeta}(-k_p^\zeta(\zeta - \zeta^*) - k_d^\zeta\dot{\zeta} + d_\zeta) + k_p^\zeta(\zeta - \zeta^*)\dot{\zeta} && (8.18)\\
&= -\dot{\zeta}k_p^\zeta(\zeta - \zeta^*) - \dot{\zeta}k_d^\zeta\dot{\zeta} + \dot{\zeta}d_\zeta + k_p^\zeta(\zeta - \zeta^*)\dot{\zeta} && (8.19)\\
&= -k_d^\zeta\dot{\zeta}^2 + \dot{\zeta}d_\zeta && (8.20)\\
&\leq \dot{\zeta}d_\zeta && (8.21)
\end{aligned}$$

Choosing input $u = d_\zeta$ and output $y = \dot{\zeta}$ yields

$$\dot{V} \leq y^T u \tag{8.22}$$

And thus the control system is passive with input $u = d_\zeta$ and output $y = \dot{\zeta}$.

According to [42], asymptotic stability is connected to passivity by zero-state observability. Given that $u = 0$ and $y = 0$, the following implications can be made

$$y = 0 \implies \dot{\zeta} = 0 \tag{8.23}$$

$$u = 0 \implies -k_p^\zeta(\zeta - \zeta^*) - k_d^\zeta \dot{\zeta} = 0 \implies \zeta = \zeta^* \tag{8.24}$$

Hence, the only solution to $\dot{\boldsymbol{\zeta}} = f(\boldsymbol{\zeta}, 0)$ that can stay identically in $\mathcal{S} = \{\boldsymbol{\zeta} \in \mathbb{R}^2 | y = 0\}$ is $\boldsymbol{\zeta}(t) \equiv \begin{bmatrix}\zeta^* & 0\end{bmatrix}^\top$, where $\boldsymbol{\zeta} = \begin{bmatrix}\zeta & \dot{\zeta}\end{bmatrix}^\top$. Thus the system is zero-state observable for $\boldsymbol{\zeta}^* = \begin{bmatrix}\zeta^* & 0\end{bmatrix}^\top$, and by extension $\boldsymbol{\zeta}^*$ is also globally asymptotically stable (globally since the storage function V is radially unbounded).

The reason why the passivity property is desirable, is because in practice these systems have shown a strong resistance to noise. Since both the environment and drone are passive, this means that the interaction between them will also be passive [42]. In addition, the summary paper of the AEROARMS project [5] mentions passivity based controllers as a good way to tackle the trade-off between performance and stability, and also allow for stable control via teleoperation with potential time delay.

8.2 Results

In chapter 9 the full 3D controller will be presented, based on the same methods used in this chapter. However, the results from the two different models are similar, so only the main points will be presented in this section, while section 9.2 will provide a more thorough discussion.

From fig. 4.4 it can be seen that the controller is successfully able to maintain a stable contact force of 3.5 N. However, simulations also showed that the controller also was capable of handling stronger interaction forces, but this requires the probe angle to be adjusted approximately to the steady-state pitch angle for a stable interaction.

The initial contact forces contains small oscillations, due to the idealistic simulation of the interaction model as a pure spring. This causes a non-physical, transient behavior. However, realistic simulations of this interaction model is

difficult, and since this is only used as an initial validation, it will be prioritized here. A more realistic interaction will be simulated in part Appendix Part I, when the system is implemented in a physics engine. In fig. 4.5 the equivalence of simulating the compliant behavior in the probe and the interaction model becomes apparent in the steady-state. The drone appears to be penetrating the wall as a result of the compliant environment.

As mentioned, the mass of the probe was omitted from the design of the controller, but included in the simulator. The validity of this assumption was validated by the simulation, and the controller has no problem compensating for it.

One of the challenges using this approach is the need to manipulate the set-points to adjust the behavior of interaction. In this simulation, the simple technique of gradually increasing the set-point to a point about 2 m beyond the inspection surface is applied, yielding good results.

Another disadvantage in this simulation is the lack of realistic actuator simulation. This causes a very quick response in the attitude dynamics when input is applied, as can be seen in the plot of θ in fig. 4.4 when the drone detaches from the wall. To counter this problem, a realistic motor simulation was added to the 3D simulations, which will be presented in the next chapter.

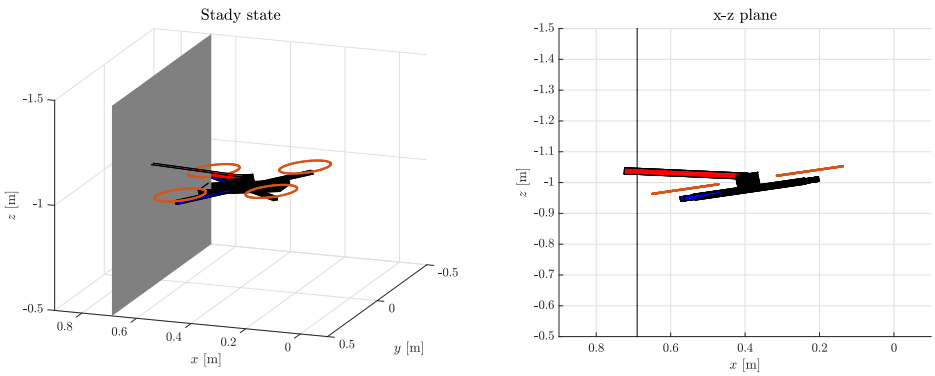


Figure 8.2: Visualization of the steady state interaction with the inspection surface

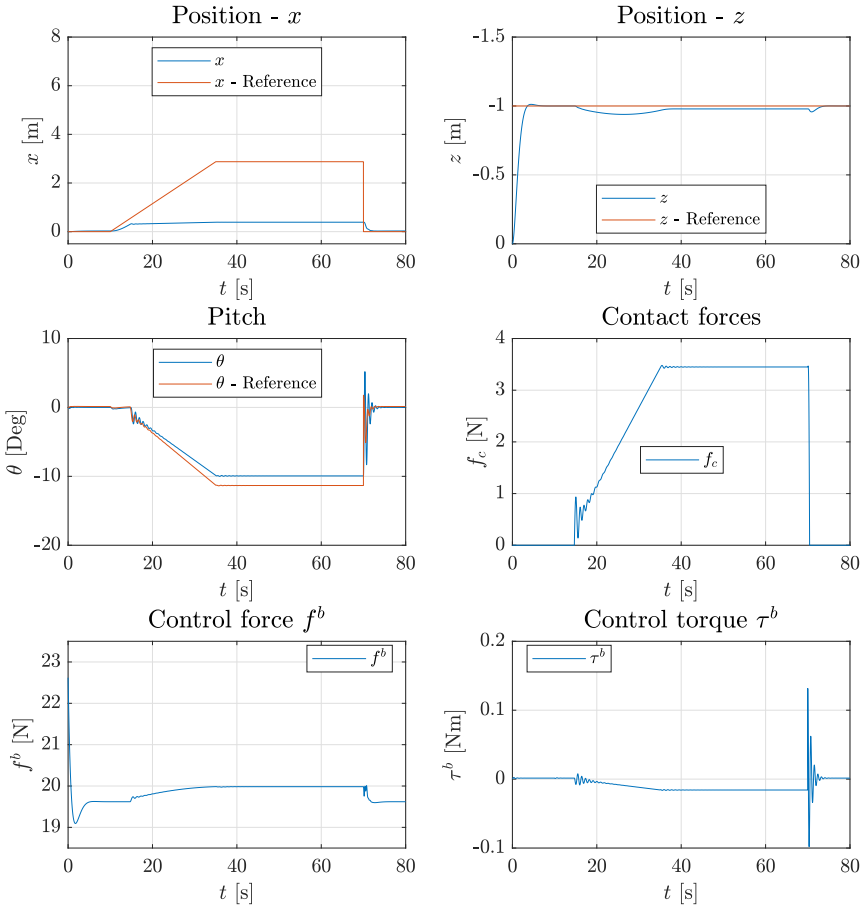


Figure 8.3: State variables, contact forces and control variables during simulation

Chapter 9

Controller 3D

This chapter will present an impedance based controller for the full 3D model of the drone, inspired by the derivations in [33]. Afterwards, simulation results are presented along with discussions.

9.1 Controller Derivation

Building on the same ideas and principles as for the 2D model, it is possible to develop a controller for the full 3D model.

The first assumption that is made is that a high-gain attitude controller implemented, such that $\mathbf{\Theta} \approx \mathbf{\Theta}^*$. This is a common assumption and also fits well with the interface to low-level controllers such as the Pixhawk[1]. Most UAV platforms comes with such an attitude control out-of-the-box. Another, more advanced, attitude controller is presented in [48].

By making this assumption it is possible to define new virtual inputs $\mathbf{\Theta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\top$ and f, such that $u = \begin{bmatrix}\phi & \theta & \psi & f\end{bmatrix}^\top$, and eq. (5.20) simplifies to

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ m_{uav}\dot{\mathbf{v}} &= m_{uav}\mathbf{g} + \mathbf{R}_b^n(\Theta)\mathbf{f} + \mathbf{f}_e \end{aligned} \tag{9.1}$$

[1] http://pixhawk.org/

$\mathbf{F}$ is first defined as [33]

$$\mathbf{F} = m_{uav}\mathbf{g} + \mathbf{R}_b^n(\Theta)\mathbf{f} \tag{9.2}$$

It is important to note that this is well defined, as long as $|\mathbf{F} - m_{uav}\mathbf{g}| > 0$. This means that the mapping is valid for all configurations except free fall, which is a scenario that is far from the operating points of the UAV as discussed in chapter 8.

Rewriting eq. (9.1) using $\mathbf{F}$ yields

$$\dot{\mathbf{p}}_b^i = \mathbf{v}^i \tag{9.3}$$

$$m_{uav}\dot{\mathbf{v}}^i = \mathbf{F} + \mathbf{f}_e \tag{9.4}$$

where according to eq. (5.21)

$$\mathbf{f}_e = \mathbf{R}_b^i\mathbf{R}_m^b\mathbf{f}_c^m \tag{9.5}$$

Then, as in [33], let the control law be given

$$\mathbf{F} = -\mathbf{K}_p(\mathbf{p}_b^i - \mathbf{p}_b^{i*}) - \mathbf{K}_d\mathbf{v}^i \tag{9.6}$$

where $\mathbf{K}_p, \mathbf{K}_d \succ 0$ are the gain matrices and $\mathbf{p}_b^{i*}$ is the desired position. As with the 2D controller, the impedance equivalence is made apparent by rewriting the system as

$$m_{uav}\ddot{\mathbf{p}}_b^i + \mathbf{K}_d\dot{\mathbf{p}}_b^i + \mathbf{K}_p(\mathbf{p}_b^i - \mathbf{p}_b^{i*}) = \mathbf{f}_e \tag{9.7}$$

This equivalence can be illustrated in the same way as before (fig. 8.1); however, this is a 3 DOF impedance with the possibility of cross-coupling (as long as $\mathbf{K}_p, \mathbf{K}_d \succ 0$). By tuning $\mathbf{K}_p$ and $\mathbf{K}_d$ it is possible adjust the stiffness and damping of the equivalent impedances reaction to the disturbance $\mathbf{f}_e$. This tuning is not trivial for the general case of an unknown, possibly unbounded, disturbance $\mathbf{f}_e$. However, only the specific scenario of docking to the inspection surface for thickness measurements are considered, which means that $\mathbf{f}_e$ will be similar for each run and the controller can be tuned to robustly handle the forces during these interactions.

Passivity Proof

Again, this controller can also be interpreted as a passivity based controller, and the following will present a proof of passivity.

The storage function $V(\mathbf{v}^i, \mathbf{p}_b^i)$ is defined as [33]

$$V(\mathbf{v}^i, \mathbf{p}_b^i) = \mathcal{K}(\mathbf{v}^i) + \mathcal{P}(\mathbf{p}_b^i) \tag{9.8}$$

where $\mathcal{K}(\mathbf{v}^i)$ is the kinetic energy and $\mathcal{P}(\mathbf{p}_b^i)$ the potential energy defined as

$$\mathcal{K}(\mathbf{v}^i) = \frac{1}{2} m_{uav} (\mathbf{v}^i)^T \mathbf{v}^i \tag{9.9}$$

$$\mathcal{P}(\mathbf{p}_b^i) = \frac{1}{2} (\mathbf{p}_b^i - \mathbf{p}_b^{i*})^T \mathbf{K}_p (\mathbf{p}_b^i - \mathbf{p}_b^{i*}) \tag{9.10}$$

Such that $V \geq 0$.

Differentiating along the trajectory yields

$$\begin{aligned}
\dot{V}(\mathbf{v}^i, \mathbf{p}_b^i) &= m_{uav} (\mathbf{v}^i)^T \dot{\mathbf{v}}^i + (\mathbf{p}_b^i - \mathbf{p}_b^{i*})^T \mathbf{K}_p \dot{p}_b^i && (9.11)\\
&= (\mathbf{v}^i)^T (\mathbf{F} + \mathbf{f}_e) + (\mathbf{p}_b^i - \mathbf{p}_b^{i*})^T \mathbf{K}_p \mathbf{v}^i && (9.12)\\
&= (\mathbf{v}^i)^T (-\mathbf{K}_p (\mathbf{p}_b^i - \mathbf{p}_b^{i*}) - \mathbf{K}_d \mathbf{v}^i + \mathbf{f}_e) + (\mathbf{p}_b^i - \mathbf{p}_b^{i*})^T \mathbf{K}_p \mathbf{v}^i && (9.13)\\
&= -(\mathbf{v}^i)^T \mathbf{K}_d \mathbf{v}^i + (\mathbf{v}^i)^T \mathbf{f}_e && (9.14)\\
&\leq (\mathbf{v}^i)^T \mathbf{f}_e && (9.15)
\end{aligned}$$

Choosing input $\mathbf{u}_p = \mathbf{f}_e$ and output $\mathbf{y}_p = \mathbf{v}^i$ yields

$$\dot{V} \leq \mathbf{y}_p^T \mathbf{u}_p \tag{9.16}$$

And thus the control system is passive with input $\mathbf{u}_p = \mathbf{d}$ and output $\mathbf{y}_p = \mathbf{v}^i$.

Again, zero-state observability [42] is proven by letting $\mathbf{u}_p = 0$ and $\mathbf{y}_p = 0$. This gives

$$\mathbf{y}_p = 0 \implies \mathbf{v}^i = 0 \tag{9.17}$$

$$\mathbf{u}_p = 0 \implies -\mathbf{K}_p (\mathbf{p}_b^i - \mathbf{p}_b^{i*}) = 0 \implies \mathbf{p}_b^i = \mathbf{p}_b^{i*}, \quad \text{since } \mathbf{K}_p \succ 0 \tag{9.18}$$

Hence the only solution to $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, 0)$ that can stay identically in $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^6 | \mathbf{y}_p = 0\}$ is $\boldsymbol{x}(t) \equiv \begin{bmatrix} \mathbf{p}_b^{i*} & 0 \end{bmatrix}^\top$, where $\boldsymbol{x} = \begin{bmatrix} \mathbf{p}_b^i & \mathbf{v}^i \end{bmatrix}^\top$. Thus the system is zero-state observable for $\boldsymbol{x}^* = \begin{bmatrix} \mathbf{p}_b^{i*} & 0 \end{bmatrix}^\top$, and by extension $\boldsymbol{x}^*$ is also globally asymptotically stable (globally since the storage function V is radially unbounded) [42].

See section 8.1 for the discussion of why the passivity property is desirable.

Generating Desired Attitude

In order to get the input to the low level attitude controller, the desired force $\mathbf{F}$ must be converted to desired Euler angles ϕ_d , θ_d and ψ_d. This section outlines this process. Since a multirotor is able to produce forces in x- and y-direction no matter the yaw angle, the desired yaw angle ψ_d can be found independently of the desired force. By first rotating $\mathbf{F}$ around the z-axis with the current yaw ψ, the body-oriented forces $\mathbf{f}^b$ are found. This can be used to find the roll- and pitch angles as follows

$$F_x^b = -\left|\mathbf{f}^b\right| \cos\phi \sin\theta \tag{9.19}$$

$$F_y^b = \left|\mathbf{f}^b\right| \sin\phi \tag{9.20}$$

$$F_z^b = -\left|\mathbf{f}^b\right| \cos\phi \cos\theta \tag{9.21}$$

By first solving eq. (9.21) for $f = \left|\mathbf{f}^b\right|$ using the current values for ϕ and θ, then solving eqs. (9.19) and (9.20) for ϕ and θ, desired angles can be obtained. However, by assuming low vertical accelerations this can be further simplified. The desired roll and pitch angles can then be found as

$$\theta_d = \text{atan2}\left(-F_x^b, m_c g\right) \tag{9.22}$$

$$\phi_d = \text{atan2}\left(F_y^b \cos\theta_d, m_c g\right) \tag{9.23}$$

These angles can then be used as input to the low-level attitude controller.

9.2 Results

From fig. 5.3 it can be seen that the controller is successfully able to maintain a stable contact force of 3.5 N. Simulations also showed that the drone was able to handle interaction forces up to 5 N without noticeable problems. Beyond this the pitch angle and interaction forces becomes large, and only small disturbances causes the drone to loose control, especially for the yaw angle.

From the plot of the contact forces in fig. 5.3, it can be seen that the initial contact forces contains small oscillations. This is caused by the idealistic simulation of the interaction model as a pure spring, and is not the best representation of a real world simulation. Similar to the 2D model, the equivalence of simulating the compliant behavior in the probe and the interaction model can be seen in fig. 5.4,

as the drone appears to be penetrating the wall as a result of the compliant environment.

Also, the 3D position controller is very dependent on the set-points of the desired position. If the point is set far beyond the inspection, the drone will get into a situation with large interaction forces, causing too much stress on the compliance device. The same technique where, a ramp function to gradually increase to the desired position at a point about 1.5 m into the wall, is applied here. From fig. 5.3 it can be seen that this method causes no large spikes in the interaction forces during the initial contact, and enables the drone to get into a good and stable contact with the environment. However, simulations done using large steps in set-points show that this causes a bouncing behavior, as the torque created by the initial contact pitches the drone backwards and away from the inspection surface. This shows that care must be taken when choosing the set-points to avoid large interaction forces.

There exist augmentations to the way of choosing set-points, for example [31]. Here the desired set-point is modeled as a dynamic system that is coupled with the actual position of the drone. The idea is that the dynamics of the desired position guides the position of the drone towards the set-point of the user. However, if the drone is unable to continue (i.e. the drone is in contact with the environment) the dynamics of the desired point will also stabilize. This is done to avoid high interaction forces as a result of large velocities, and avoid scenarios where the desired position is far outside the area constrained by the environment. Depending on the specific application of the controller, this could be a potential solution if the current method of set-point creation is inadequate.

The mass of the probe and the moments created by its gravity was omitted from the modeling of the controller; however, it is included in these simulations. The results validates the assumption that the mass of the probe has little to no influence on the behavior of the drone.

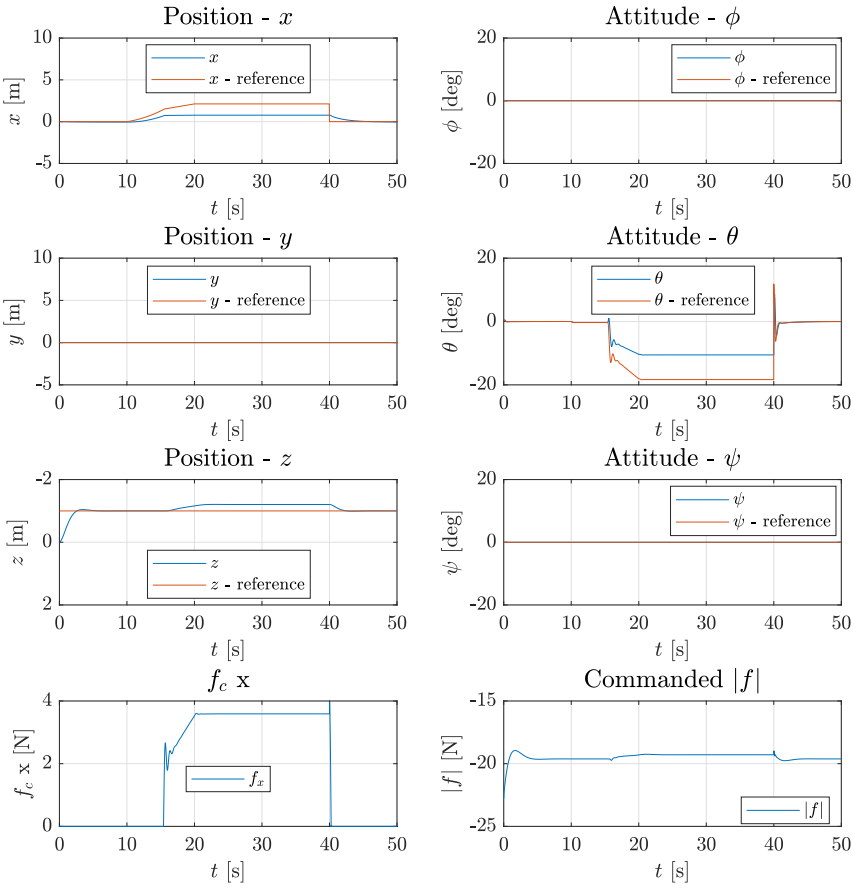


Figure 9.1: State variables, contact forces and control variables during simulation

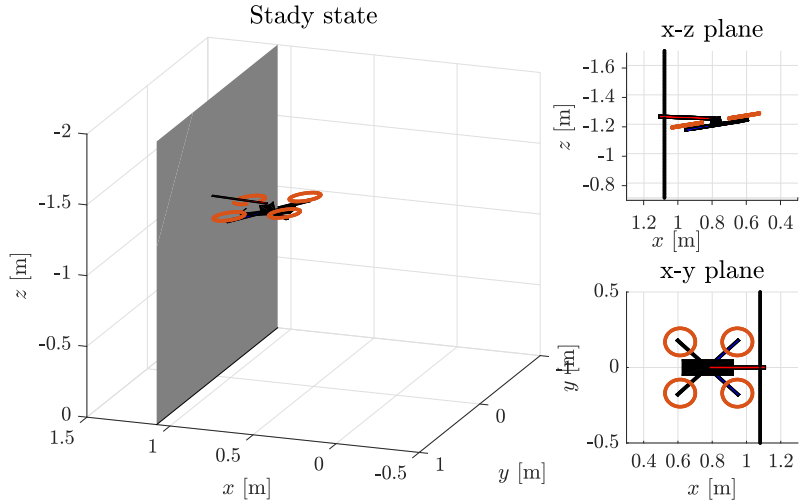


Figure 9.2: Visualization of the steady state interaction with the inspection surface

The disturbances in the attitude Θ caused by the moments from the interaction forces were explicitly addressed in the controller derivation, and the assumption was made that a low level attitude controller was implemented such that $\Theta \approx \Theta_d$. The plot of θ in fig. 5.3 shows that this assumption is violated; however, as a result of the impedance behavior of the drone this does not create any large problems, as the drone stabilizes with constant deviation in both attitude and position. However, the desired total thrust is calculated based on the assumption that $\Theta \approx \Theta_d$, but since $|\theta| < |\theta|_d$ the desired vertical thrust becomes a little to large. This causes the drone to rise slightly above the desired set-point as seen by the z-position in fig. 5.3. The larger the interaction forces are, the larger this problem becomes, and this is a contributing factor to destabilization in the scenarios with strong contact forces. However, the impedance behavior in the z-direction helps to stabilize this, as it tries to compensate for the effect by lowering the vertical thrust. Another solution to minimize these errors is to use the actual pitch and roll angles to compute the desired force, and not the desired ones. (When the simplified method in section 9.1 is used, the desired angles are used to find f)

From the results presented here it is clear that one of the main drawbacks of

this controller is the adjustment of the impedance parameters. Even though the impedance of the drone is tuned, it is not obvious how this affects the contact forces. This is especially true for the transient phase, where high peaks in interaction forces could cause violation of the assumptions of the controller. An immediate idea to solve this non-trivial relationship between position and force, is to rather create a velocity-controller, as this has a more direct connection to the forces experienced during interaction. However, such a controller also neglects to directly address an important part of the scenario, namely the attitude during the interaction. The coupling between velocities and attitudes are generally not trivial, as the drone might for instance have a large velocity in the x-direction while the pitch angle is still zero. Hence, if more fine-grained control of the interaction is required, a better solution would be to look into creating trajectories, giving both the desired velocities and attitudes required to facilitate a stable interaction. The next chapter will look into creating such trajectories.

This simulation is used as an initial validation of the controller, and only one scenario is considered. In part Appendix Part I the controller is implemented in ROS/Gazebo, which represents a more accurate simulation of the drone and interaction. Hence, this part will present several scenarios and a more thorough discussion of the qualities of the controller (see chapter C), before the implementation and testing of the controllers on the hardware platform (part VII).

To summarize, the largest drawbacks of using the impedance based position controller is the need for set-point manipulation and accurate impedance tuning to ensure stable interaction. However, for set-point manipulations simple techniques, such as a ramp function or inserting an intermediate set-point, seems to suffice. Beyond this, the impedance controller is able to ensure stable interaction in simulations and it is a desirable solution due to its simplicity.

Chapter 10

Tracking Of Pre-computed Optimal Trajectory

This chapter will present a different approach to the problem, by looking at pre-generating trajectories for the interaction. This chapter will first present the motivation behind this approach, before giving the necessary background theory. Thereafter, the UAV interaction problem is formulated in this framework, before results and discussions are presented.

10.1 Motivation

The controllers mentioned in the chapters above provides convenient passivity proofs and constitutes a controller that is capable of performing free-flight maneuvers and interaction tasks simultaneously. However, the performance of the controller during interaction heavily depends on the choice of the set-point $\mathbf{p}_b^{*i}$. This is not a problem if the probe and compliance device is designed robust enough to handle all relevant interactions, but this is not always the case. The drone might be dependent on a specific entry angle θ in the initial phase of the interaction to make sure that the probe comes in sufficient contact to get a valid measurement. Entry velocity also plays a major role, as a large entry velocity will result in a bouncing behavior in the best case, or a damage to the compliance device or sensor in the worst case.

While the simplicity of the position controller is appealing, a more advanced framework might be necessary. As a solution to the drawbacks mentioned above, a more problem-oriented solution would be to generate a trajectory for the interaction with the environment. This would allow for tuning entry velocity and pitch angle of during the impact, to facilitate the best possible change of a successful interaction. One solution is to use on-line Model predictive control (MPC) to control the drone, similar to what is done in [3]. However, this requires a lot of on-board computation power, which might not be available on the drone platform described in this thesis. The drone platform is running other important applications that require computational resources, rendering MPC as an undesirable solution strategy.

Another approach to this problem would be to pre-generate a trajectory from a certain set-point from the inspection surface. During operations the drone would fly to this set-point, before trying to execute the given maneuver. During the trajectory following it is possible to use different control mechanisms such as LQR or PD trajectory control. This could also facilitate the creation of an abort system for more robust and safe interaction, as the drone would abort the interaction attempt and retry if it deviates too much from the pre-generated trajectory. This approach will give a more fine-grained control over the interaction as a whole, compared to the position controller. However, as discussed in section 7.1, a controller that strictly enforces the trajectory during interaction, without considering unknown forces, might cause a large build-up of interaction forces and cause breakage.

Physics engine simulations presented in section 10.5 prove that it is possible to get the desired interaction; however, the experimental validation of the following controller is left as future work.

10.2 Background

This section will introduce the basic of optimization for dynamic models. For a more thorough treatment of the subject, the reader is referred to [53] for general optimization and [28] for optimization from a control perspective. The following is based on these two books.

10.2.1 Optimization Problems

A subset of optimization problems are called Quadratic Programming (QP) problems. These can in general be formulated as [53]

$$\min_{\mathbf{z}\in\mathbb{R}^n} f(\mathbf{z}) = \mathbf{z}^\top \mathbf{Q}\mathbf{z} + \mathbf{d}^\top \mathbf{z} \tag{10.1}$$

subject to

$$c_i(\mathbf{z}) = \mathbf{a}_i^\top \mathbf{z} - b_i = 0, i \in \mathcal{E} \tag{10.2}$$

$$c_i(\mathbf{z}) = \mathbf{a}_i^\top \mathbf{z} - b_i \geq 0, i \in \mathcal{I} \tag{10.3}$$

where $\mathcal{E}$ is the equality constraints, and $\mathcal{I}$ are the inequality constraints.

If Q is positive semidefinite, i.e., $Q \succeq 0$, then the objective function is convex.

If in addition the equality constraints are linear and c_i are convex functions for $i \in \mathcal{I}$, then the QP-problem is a convex programming problem. Convex QP-problems can be solved very efficiently, with guarantees for a globally optimal solution.

If the function $f(\mathbf{z})$ and/or the constraint functions $c_i(\mathbf{z})$ are nonlinear, then the optimization is a Nonlinear Program (NLP). If only the inequality constraints are nonlinear then the problem might still be convex, depending on the feasible region. However, nonlinear equality constraints always give rise to non-convex problems. The feasible region is defined as [53]

$$\Omega = \{\mathbf{z} \in \mathbb{R}^n | \, (c_i(\mathbf{z}) = 0, i \in \mathcal{E}) \wedge (c_i(\mathbf{z}) \geq 0, i \in \mathcal{I})\} \tag{10.4}$$

General NLP problems and non-convex problems are much harder to solve, and even with convergence there is no guarantees of a globally optimal solution.

10.2.2 Optimization of Dynamic Systems

Assume a model is given as

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t \tag{10.5}$$

Then the following QP problem can be formulated [28]

$$\min_{\mathbf{z}\in\mathbb{R}^n} f(\mathbf{z}) = \sum_{t=0}^{N-1} \frac{1}{2}\mathbf{x}_{t+1}^\top \mathbf{Q}_t \mathbf{x}_{t+1} + \frac{1}{2}\mathbf{u}_t^\top \mathbf{R}_t \mathbf{u}_t + \frac{1}{2}\mathbf{\Delta u}_t^\top \mathbf{R}_{\Delta t} \mathbf{\Delta u}_t \tag{10.6}$$

subject to

$$\mathbf{x}_{t+1} = \mathbf{A}_t\mathbf{x}_t + \mathbf{B}_t\mathbf{u}_t \tag{10.7}$$
$$\mathbf{x}_0, \mathbf{u}_{-1} = \text{ given} \tag{10.8}$$
$$\mathbf{x}^{\text{low}} \leq \mathbf{x}_t \leq \mathbf{x}^{\text{high}} \tag{10.9}$$
$$\mathbf{u}^{\text{low}} \leq \mathbf{u}_t \leq \mathbf{u}^{\text{high}} \tag{10.10}$$
$$-\boldsymbol{\Delta}\mathbf{u}^{\text{high}} \leq \boldsymbol{\Delta}\mathbf{u}_t \leq \boldsymbol{\Delta}\mathbf{u}^{\text{high}} \tag{10.11}$$

where

$$\mathbf{Q}_t \succeq 0 \tag{10.12}$$
$$\mathbf{R}_t \succeq 0 \tag{10.13}$$
$$\mathbf{R}_{\Delta t} \succeq 0 \tag{10.14}$$
$$\boldsymbol{\Delta}\mathbf{u}_t = \mathbf{u}_t - \mathbf{u}_{t-1} \tag{10.15}$$
$$\mathbf{z}^\top = \left(\mathbf{x}_1^\top, \ldots, \mathbf{x}_N^\top, \mathbf{u}_0^\top \ldots, \mathbf{u}_{N-1}^\top\right) \tag{10.16}$$
$$n = N \cdot (n_x + n_u) \tag{10.17}$$

To find the optimal solution in a solver such as MATLAB, this has to be formulated on standard form. This is done by using the model as equality constraints such that $\mathbf{A}_{eq}\mathbf{z} = \mathbf{B}_{eq}$ where

$$\mathbf{A}_{eq} = \left[\begin{array}{ccccc|ccccc} \mathbf{I} & 0 & \cdots & \cdots & 0 & -\mathbf{B}_0 & 0 & \cdots & \cdots & 0 \\ -\mathbf{A}_1 & \mathbf{I} & \ddots & & \vdots & 0 & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 & \vdots & & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -\mathbf{A}_N & \mathbf{I} & 0 & \cdots & \cdots & 0 & -\mathbf{B}_N \end{array}\right] \tag{10.18}$$

and

$$\mathbf{b}_eq = \begin{bmatrix} \mathbf{A}_0 x_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10.19}$$

The $\mathbf{Q}$ and $\mathbf{R}$ matrices are also put into a block diagonal form. For more details see [28].

10.3 Optimal Trajectory for UAVs

Using the model for the dominant multirotor dynamics from section 9.1

$$\begin{aligned} \dot{\mathbf{p}} &= \mathbf{v} \\ m_{uav}\dot{\mathbf{v}} &= m_{uav}\mathbf{g} + \mathbf{R}_b^n(\Theta)\mathbf{f} \end{aligned} \tag{10.20}$$

This is a non-linear model, and hence would give raise to an NLP optimization problem. However, the trick is to realize that it is not necessary to include the attitude in the optimization problem, and by defining

$$\mathbf{F} = m_{uav}\mathbf{g} + \mathbf{R}_b^n(\Theta)\mathbf{f} \tag{10.21}$$

the resulting model is linear, which is far more suited to optimization on ($\mathbf{F}$ is the new input):

$$\dot{\mathbf{p}}_b^i = \mathbf{v}^i \tag{10.22}$$

$$m_{uav}\dot{\mathbf{v}}^i = \mathbf{F} \tag{10.23}$$

The desired angles can be generated from $\mathbf{F}$ by utilizing the technique described in section 9.1. A potential pitfall using this method is that the trajectories will not become feasible to perform, as the attitude dynamics are not considered and $\mathbf{F}$ could have rapid changes in direction and magnitude. To overcome this, a penalty was added to the changes in each component of $\mathbf{F}$ (tuning $\mathbf{R}_{\Delta t}$), to facilitate the generation of feasible trajectories.

This model was then converted to the standard form described in section 10.2.2 and solved using MATLAB. The desired force against the inspection surface can be implemented by constraining the last input $\mathbf{u}_{N-1}$ to produce the desired force. Other limitations, such as no horizontal velocity in the direction parallel to the surface can also easily be implemented.

10.4 Trajectory Generation Results

In this experiment the inspection surface was placed such that the drone would come in contact with it when it reaches $\mathbf{p} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$ and with θ equal to the probe angle. This was done by constraining the last state $\mathbf{x}_n$ to the point of the Center of Gravity (COG) of the drone during impact. Penalty was added to $\mathbf{R}_t$

for using forces in x and y directions; however, no penalty was added in the z direction to allow the drone to use the necessary force to maintain altitude. $\mathbf{Q}_t$ was set such that all states were penalized equally, but also smaller relative to $\mathbf{R}_t$ such that the optimal solution would include a low approach velocity. $\mathbf{R}_{\Delta t}$ was created to penalize changes in the input forces as discussed in the section above. The desired interaction force was set to $u_{N-1,x} = 4\,\text{N}$. The time horizon was set to 3 s, with a time step of 0.1 s.

The resulting trajectories are shown in fig. 10.1, where the attitudes required to generate the desired forces are shown in the bottom left. Only θ is shown, as the trajectory depicted takes place in the xz-plane. Figure 10.2 shows the trajectory of the drone with faded colors and the end pose in solid.

Both figs. 10.1 and 10.2 shows that trajectory created constitutes a path with reasonable approach velocity, and that the attitude changes to get the correct force during interaction towards the end off the defined time horizon.

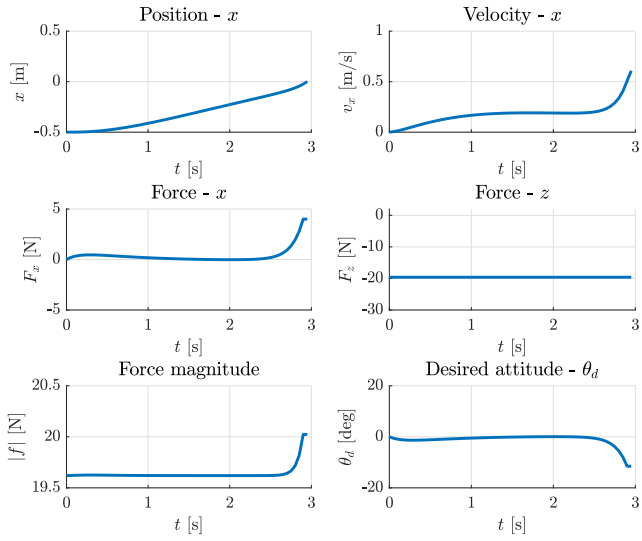


Figure 10.1: Solution to the optimization problem

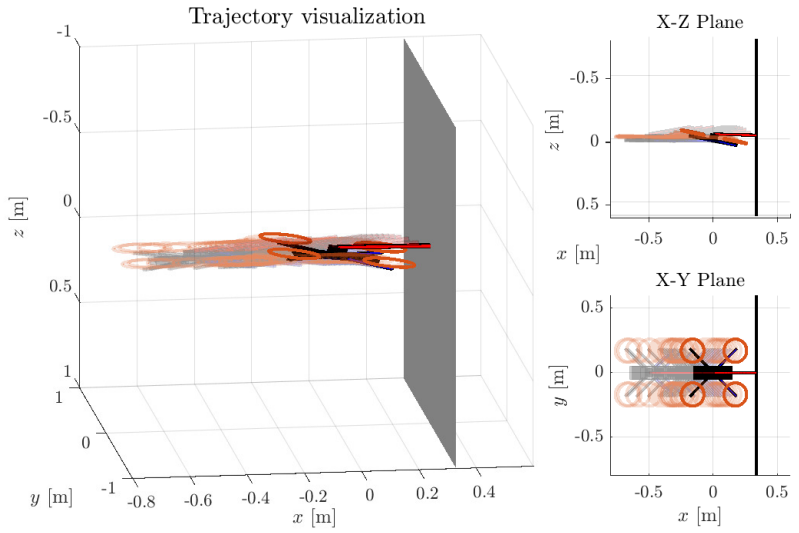


Figure 10.2: Visualization of the optimal trajectory

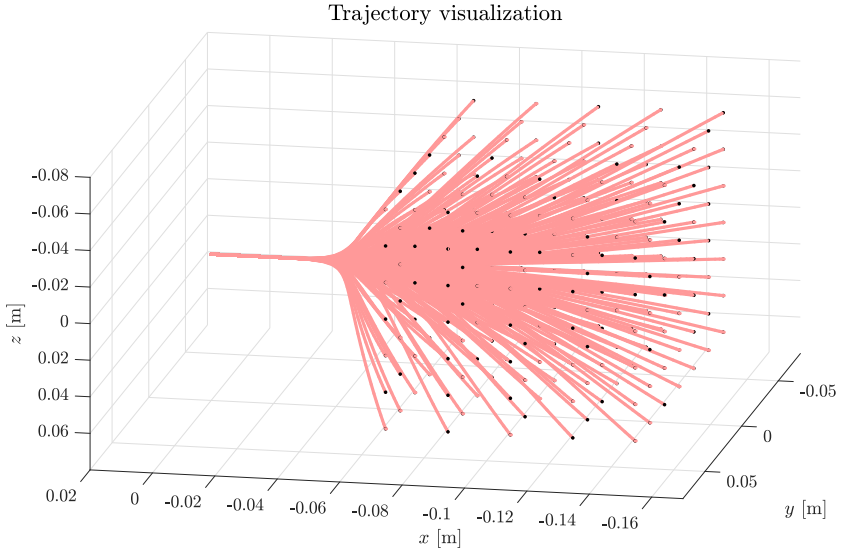


Figure 10.3: Visualization of cone of optimal trajectories

Figure 10.3 shows how this framework can be used to generate paths from a wide range of different starting positions. The velocity parallel to the surface was constrained to be zero for the last time steps, and the figure shows how all trajectories converge to one common path the final approach towards the wall.

10.5 Simulation Results

In this section, the tracking of the generated trajectory was implemented in a physics engine framework, as detailed in part Appendix Part I. The simulation below shows a pure feed-forward of the attitude, and the reference attitudes was passed directly to the low-level attitude controller.

The reader is strongly recommended to watch the video of the simulation by following the QR-code or link given in fig. 10.4. This will give a more intuitive understanding of the results presented in fig. 10.5 and fig. 10.6.

Figure 10.4: QR-code for the simulation (https://youtu.be/DetFfxNkz9g)

During free-flight, the generated trajectory is tracked well by the low-level attitude controller. However, during the impact, there are clear deviations from the trajectory. The trajectory was constrained to hold a constant force towards to surface; however, due to the complex constraints imposed by the environment, this interaction is far from constant. While the results are expected, they also show the biggest drawback of this solution, as it does not give an explicit way to handle these deviations. However, as mentioned before, it is important for any controller not to cause build-up of interaction forces, as this could lead to breakage.

After the initial transient effect from the compliance device in the probe has settled, the force towards the surface is stable around 3.5 N, as see in the top graph in fig. 10.5. This is very similar to the results in chapter C from the position based controllers developed in chapter 9. The behavior of the yaw angle during interaction is also similar for the two controllers.

To summarize the simulation results, the method described in this chapter shows promising results to give more accurate control of the interaction than the position based controllers provide. However, more advanced tracking methods should be implemented in order to properly evaluate the full potential of the method.

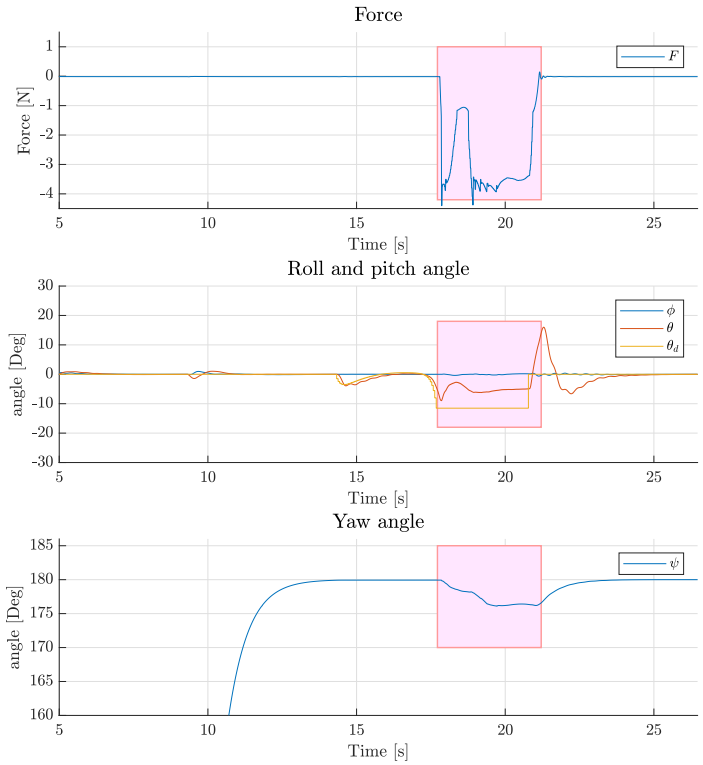


Figure 10.5: Attitude and force for simulation with feed-forward control of trajectory. (Red area shows the duration of contact)

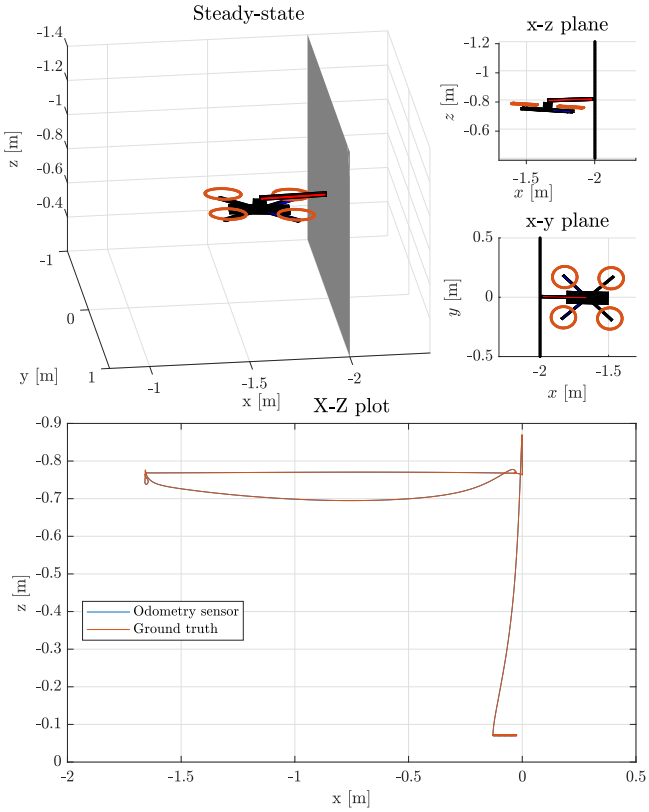


Figure 10.6: Trajectory simulation.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

10.6 Discussion

To implement these trajectories on the actual drone, a trajectory controller is beneficial. Since the path is pre-generated, it is possible to feed-forward the desired velocities and accelerations along the path to get a more precise tracking of the trajectory. This was not done in the results shown in the previous section,

and such controller could improve the tracking further. As mentioned in the beginning of this chapter, the error correction between the actual and desired path can be done by a PD-controller, or using more advanced techniques like LQR.

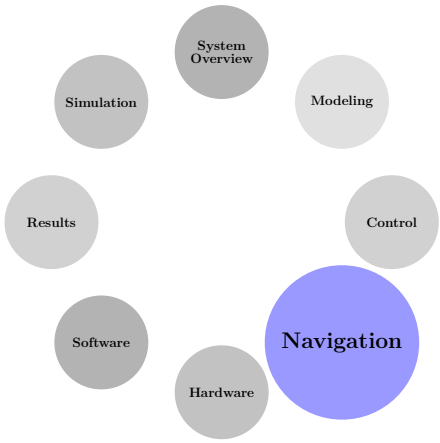
Another approach related to the ideas of the above solutions, is the use of explicit MPC [8]. The idea is to solve the QP-problem off-line for all $x \in \mathbb{R}^n$ to find en explicit control law $u = u(x)$. The technique uses a multiparametric QP algorithm, and the resulting control law is a continuous piecewise affine function. The benefit of this approach is that it eliminates the need to solve the optimization problem on-board; however, with the drawback of the additional storage space required.

The trajectories created are clearly feasible maneuvers for the drone, as demonstrated by the above simulations. The benefit of this over the position controller approach, is that it captures more of the essential dynamics prior to the impact. While the position controller can be proven stable in steady state, the trajectory allows for detailed tuning of the initial impact. This allows to better relate the attitude, velocity and force against the inspection surface, and allows for planning and control at a more precise level. However, a tracking controller needs to address the deviations that occur during impact in a way that does not cause build-up of interaction forces in order to ensure stable interaction.

To summarize, the position based controllers provides proof of stable interaction in the contact phase of the interaction, while they lack explicit control of the approach phase. The trajectory controller does, however, allow for more tuning of the approach phase, but fails to address the stability during the contact. The priority of stability during contact supersedes most other performance criteria, and in addition the approach phase for the impedance based position controllers can be implicitly controlled by set-point manipulations, as discussed in section 9.2. Hence, this was the method chosen to be implemented on the hardware platform. In part VII the performance of the position based controllers developed in chapter 5 are shown to be satisfactory for successful docking in flight tests, and thus the implementation and experimental validation of the trajectory tracking on a hardware platform has been left as future work.

Part IV

Navigation



Chapter 11

Background

This chapter will first define and motivate the necessary concepts for indoor navigation and clarify the scope of the navigation solutions developed in this thesis. Thereafter, relevant theory for the upcoming chapters is presented. After a discussion about methods for relative heading estimation in chapter 12, chapter 13 will implement and compare methods for relative heading estimation. These are based on regression and RANSAC, and while background theory for the latter will be presented below, theory on regression is not included. However, the reader is referred to [30] for more information on the subject.

11.1 Navigation in Indoor Environments

Most drones available for commercial use require some sort of GNSS, such as the GPS to solve the drift problem in dead reckoning in inertial navigation. Another common approach to the problem is visual navigation, such as visual odometry based on camera or an optical flow sensor. However, neither of these are available in most industrial environments, as the weak GNSS signals fail to penetrate most building surfaces, and poor lighting render visual techniques useless in most situations.

There exist an extensive amount of literature on the topic of navigation, and it is common to distinguish between the following concepts [18]:

- Global navigation - Determine the position in absolute or map-referenced terms. Commonly used to move to a target destination.

- Local navigation - Determine the position relative to landmarks in the surrounding environment. Commonly used to provide information in order to interact with these objects or landmarks.

The two terms are not strictly separate, e.g. global navigation methods using the relative position to multiple local landmarks to determine its global position, such as [51]. A common misconception is that local navigation is memoryless, in the sense that it only uses the momentarily available sensor data to determine its surroundings. However, local navigation also incorporates more advanced techniques, such as smoothing of raw sensor measurements via filtering or sensor fusion, and detection of a new tracked obstacle.

This thesis is concerned with integrating autonomous thickness measurements into an existing inspection scenario. The pilot will fly to an area of interest, and request a thickness measurement. Hence, the operation is restricted to a local area, and the most critical information to successfully execute the maneuver is information about the range and relative orientation to the inspection surface of interest. In addition, range information about obstacles in the direction parallel to the inspection surface is helpful to avoid drift and sideways movement during the operation. While a robust global navigation solution would allow for fully autonomous inspections from takeoff to landing, this thesis will focus on developing local solutions in order to successfully integrate autonomous thickness measurements into a pilot assisted inspection scenario. This is to keep the scope within a reasonable range, and also the local solutions developed will provide useful information to future global methods.

As mentioned above, the orientation relative to the inspection surface in the horizontal plane is of great interest during the thickness measurement, and in the following this is called the *relative heading* with respect to the inspection surface. A common approach is to use a camera to find the heading relative to some object or feature, such as in [35]. As mentioned, this approach will not be feasible for a big part of the inspection scenarios considered in this thesis, as they will be conducted in poorly lit environments, with dust and other particle pollution. Chapter 12 will describe methods for relative heading estimation based on sensors capable of operating in these conditions.

11.2 Random Sample Consensus (RANSAC)

This section will cover the necessary background theory on Random Sample Consensus (RANSAC). For a more thorough introduction the reader is referred to the original RANSAC publication [26]. RANSAC is often used in the field of computer vision, and books such as [27] gives a good introduction to the algorithm in this context.

Given a set of observed data containing outliers, RANSAC is an iterative method that estimate parameters of a mathematical model from a set of data points. The basic assumption is that the data points come from two different groups:

1. **Inliers**: Data that can be explained/generated by some set of model parameters; however, they may be subject to noise.
2. **Outliers**: Data points that does not fit the model, such as data from extreme noise values, measurement errors or incorrect interpretation of data. These have no influence on the values of the model that generated the data.

RANSAC belongs to the set of non-deterministic algorithms as it produces a good result only with a certain probability. The probability of finding a good solution increases with the number of iterations.

A summary of the RANSAC algorithm can be given as follows [26]:
Given n data points and the error threshold e_t. For every iteration k, pick m (where $m << n$ usually) of the data points at random. Based on the random data points estimate the model M_i, and compute the distance of all the data points from the new model M_i. All data points with a distance smaller than e_t are marked as inliers, and if the current ratio of inliers are better than the current best model then M_i is set as the new best model. After k iterations the best model is returned.

Hence, RANSAC also assumes that given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data. In fig. 11.1 the data was generated by half the points coming from a line with additive Gaussian noise, while the other points were sampled randomly from a uniform distribution of the same space. The result of fitting a line model using RANSAC is seen on the left side of fig. 11.1.

The greatest advantages of the RANSAC algorithm is the robust estimation of the model parameters [40], and this holds true even with a large ratio of outliers, as demonstrated by the example in fig. 11.1.

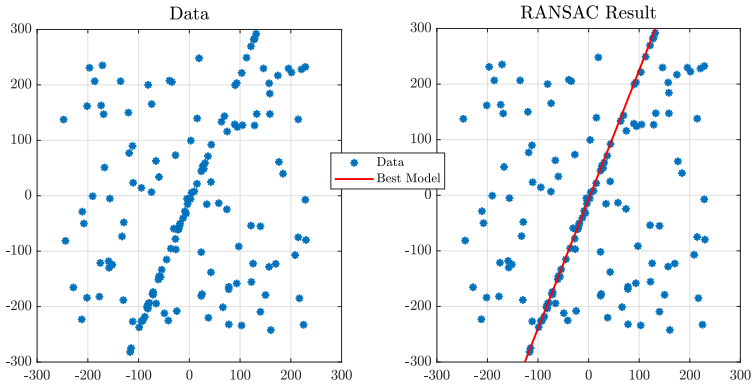


Figure 11.1: RANSAC used for line estimation on test data

A disadvantage of RANSAC is that there is no upper bound on the number of iterations required to compute the optimal solution (except exhausting all permutations of selecting m data points). Thus, since k is chosen much smaller, the solution obtained may not be optimal; or in the worst case it may fit the data poorly. Hence, RANSAC represent a trade-off between computing power (number of iterations) and the probability of a reasonable model being returned.

11.2.1 Choosing the Iteration Number

It is possible to calculate an estimate of the number of iterations k needed to produce a good model. Let w be the ratio between inliers and the total number of data points. Assuming that n points are selected *independently* each iteration for model estimation, then $1 - w^n$ is the probability that at least one of the n points is an outlier (which is assumed to results in a bad model). Let p be the probability of returning a successful model. Then the following relationship holds:

$$1 - p = (1 - w^n)^k \tag{11.1}$$

Solving for k yields

$$k = \frac{\log(1 - p)}{\log(1 - w^n)} \tag{11.2}$$

Hence, the number of iterations may be chosen as in eq. (11.2). This will be utilized when the algorithm is implemented in section 13.3.

Chapter 12

Methods for Relative Heading Estimation

This chapter will discuss different solutions for relative heading estimation, based on available sensors that are capable of functioning in poorly lit, industrial environments. The three solutions considered are based on sensors that are available to be mounted on the development drone:

- ToF sensors (section 12.1)
- RAdio Detection And Rangings (RADARs) (section 12.2)
- 2D scanning LIDAR (section 12.3)

For each of the sensors, a short introduction to the sensor technique will be given, before a solution for relative heading estimation is discussed.

It should be noted that a solution based on fusion of the methods suggested in this chapter is possible. However, for the scenario considered in this thesis, the advantages this provides is not outweighed by the disadvantage of added weight from multiple redundant sensors. Hence, only methods based on one sensor type will be discussed in the following.

12.1 ToF Sensors

Time of Flight (ToF) is a method used for measuring the distance along a line from the sensor to some object. A signal is sent from the transmitter, reflected by an object and returned back to the sensor. Different types of carrier signals can be utilized with this principle; however, the two most common are light and sound. Based on the knowledge of the speed of light or sound in the current medium, it is possible to calculate the range to the reflecting object.

The specific sensors considered in this thesis are the TeraRanger sensors, which uses infrared (IR) light (described in more detail in section 14.3). The biggest drawback of using IR light is usually the disturbance from natural ambient light. However, this is not a problem for the indoor inspection scenarios in question, but other disturbance sources might be present, such as large dust particles or highly reflective surfaces.

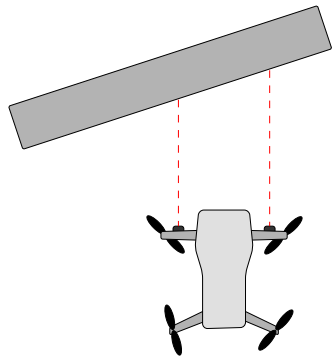


Figure 12.1: Illustration of solution using ToF

12.1.1 Solution Description

The basic principle is to use two forward facing ToF sensors separated by a baseline, to measure two different ranges. The relative heading can then be estimated by simple geometrical considerations.

Given the two range measurements r_l (left) and r_r (right), and the displacement d of the ToF from the body x-axis, the relative heading with respect to the inspection surface is given by:

$$\psi_{rel} = \text{atan2}(r_r - r_l, 2d) \tag{12.1}$$

However, if noise is present in the system we get the following measurements $\tilde{r}_d$

$$\tilde{r}_d = r_d + \epsilon_d \tag{12.2}$$

for $d \in \{l, r\}$, where $\epsilon_d \sim N(0, \sigma_d^2)$ and $N(\mu, \sigma^2)$ is a normal distribution with mean μ and variance σ^2.

Hence,

$$\tilde{r}_r - \tilde{r}_l = r_r - r_l + \epsilon_t \tag{12.3}$$

where $\epsilon_t \sim N(0, \sigma_r^2 + \sigma_l^2)$. Under the assumption that the noise has the same variance for both sensors($\sigma_r = \sigma_l = \sigma$) this simplifies to $\epsilon_t \sim N(0, 2\sigma^2)$. Hence the difference used in eq. (12.1) will be affected by a noise with twice the variance of the ToF sensors.

However, pure sensor noise is not the only problem for this solution. Small irregularities in the geometry of the surface will also directly influence the relative heading estimation. Also, if one of the sensors are unable to get a measurement at some time instances, no heading estimate will be available.

Pros:

- Simple
- Cheap

Cons:

- Poor robustness to sensor noise
- Poor robustness to surface geometry
- Poor robustness to temporary sensor failure.

In conclusion, the simplicity of the above solution is outweighed by the poor robustness against all error sources. Hence, the ToF sensor will not be chosen as the main method for relative heading estimation.

12.2 RADAR

Recent advancements in sensor technology have allowed small RADARs to be mounted on UAVs, such as in [38] and [15]. The basic principle behind RADAR, is to emit electromagnetic waves and observe the incoming echoes from surrounding objects. The specific RADAR sensor available is a XeThru X4[1], which returns both amplitude and phase information from the returning waves. Then, range information can be extracted from the amplitude, while the phase shift will reveal information about the relative velocity between the sender and reflecting object.

While RADARs provide a lot of useful information about the surrounding environment, they are also prone to a range of error sources. Examples of such are receiver noise generated by random motions of electrons at the input, clutter generated by non-important objects in the vicinity of the RADAR, and multi-path errors from signals bouncing of nearby objects. For a more thorough review and explanation of RADAR related error sources the reader is referred to [19].

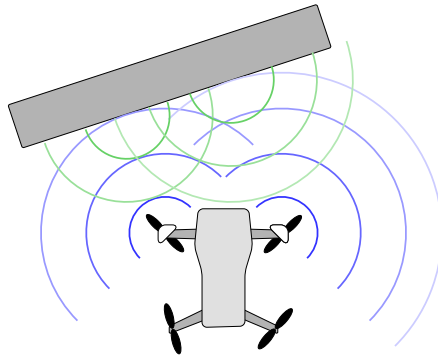


Figure 12.2: Illustration of solution using RADAR

[1] https://www.xethru.com/x4-radar-chip.html/

12.2.1 Solution Description

There are several different approaches for navigation using RADAR sensors. The simplest is using two forward mounted RADARs, and take an approach similar to the calculation method used for the ToF sensors. However, the beam width will significantly influence the results. Earlier tests conducted by Scout DI indicate that the most significant reflection is returned for the distance normal to the inspection surface, and hence it is not the distance along the sensor direction that is detected. While it is still possible to calculate the relative heading based on this, problems arise when other obstacles are nearby. Since the amplitude information is not strictly directional, this could lead to ambiguities about which range measurements are used. However, compared to the other methods discussed in this chapter the beam width also gives an advantage, as the range estimate is less affected by deviations in roll and pitch.

Amplitude and phase information from the X4 sensor is also returned in different bins based on range. Hence, the accuracy of the heading estimate is also dependent on the resolution determined by the bin size. The X4 can have bins as small as 5 mm, which provide more accuracy than most ToF sensors.

More advanced methods such as an occupancy grid [22] or Synthetic Aperture RADAR (SAR) [15] might also be possible. However, these usually require precise location information about the drone, which make them less suitable for the scenarios considered in this thesis.

Pros:

- More robust
- Can incorporate velocity information
- Work for larger deviations in roll and pitch

Cons:

- Complex signal processing
- Large minimum distance for measurement
- Unpredictable error sources
- Ambiguities in which range is measured due to beam width

In conclusion, the advanced signal processing, along with the uncertainty in the accuracy of the methods proposed above renders RADARs a less attractive choice for the purpose of this thesis.

12.3 2D Scanning Lidar

The last option considered is to use a 2D scanning LIDAR. This solution features a spinning laser that measures ranges around the entire sensor. An illustration of the principle is shown in fig. 12.3. 3D LIDAR exists, but the amount of data generated and the computing power required to process this data leaves this as an undesirable method on the hardware used on the development drone. The specific LIDAR available is an RPLIDAR A2, and will be described in detail in section 14.6.

One drawback of using a scanning LIDAR is timing issues. Since the laser is spinning, there will be a time delay between each successive scan points. In cases where high precision is needed, this means that every scan point needs to be rotated into some common reference system using the precise attitude at the exact moment of measurement. This requires very precise synchronization of attitude estimates and laser scans.

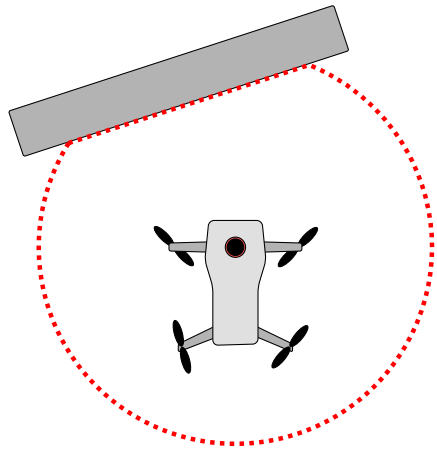


Figure 12.3: Illustration of solution using LIDAR

12.3.1 Solution Description

The basis of the solution is using the range measurements provided to estimate the inspection surface as a line. When the line has been found, the relative heading and range can be found with respect to the estimated line. Several techniques exist for line estimation based on data points, and the next chapter will present solutions based on regression and RANSAC.

It should be noted that both the solution based on ToF and LIDAR are dependent on the roll and pitch angle. If this assumption is severely violated, it is difficult to interpret the data in any meaningful way. However, since the contact based inspection scenarios will be performed with low horizontal velocities and hence small displacements in roll and pitch, this drawback will not be significant in most situations.

More advanced navigation techniques for a 2D LIDAR are possible, such as Simultaneous Location And Mapping (SLAM)[46]. However, the fact that multirotors do not operate in a single plane, such as ground robots, will affect the robustness of the solutions.

Pros:

- More robust to sensor noise
- More robust to surface geometry
- More robust to temporary, spatial sensor failure
- Easier signal processing than RADAR based solutions

Cons:

- More complex data processing than ToF sensors
- Poor performance on highly reflective or absorptive surfaces
- Time synchronization for data points needs to be solved for methods requiring high precision
- Dependency on pitch and roll angle

At the core, the methods based on ToF sensors and LIDAR both operate on detected points on the inspection surface. Further, both methods attempts the find a line based on the points provided; however, the LIDAR provides more data points than the minimum of two needed to create a line model. This gives increased robustness to noisy sensor measurement and provides some redundancy in case of a faulty measurement. It also provides some robustness against dents and bumps in the surface geometry, as the line can be estimated based on points in other regions.

Based on the pros and cons presented for the different sensors, the decision was made to use LIDAR as the main navigation sensor for range and relative heading estimation. The main arguments were that the additional data points from the LIDAR enables a more robust solution compared to using ToF sensors. The LIDAR solution also offers easier signal processing and data interpretation compared to RADARs, without obvious compromise in the robustness of the overall solution.

Chapter 13

Implementation and Evaluation

This chapter will develop, implement and compare different solutions for relative heading estimation based on data from a 2D scanning LIDAR. First, a method for adaptive selection of a section of interest is discussed, before surface detection methods based on regression and RANSAC are presented. Thereafter, the results from experimental validation is presented. While the visualizations in this chapter was done using Matlab, the implementations have been done in the DUNE framework, described in part VI.

It should be noted that when the ultrasonic measurement probe and the LIDAR are mounted on the drone, some of the vision in front LIDAR is obstructed. The sector affected by this is roughly $\pm 4°$ in front of the drone. Hence, the methods developed in this thesis should perform satisfactory within these restrictions. The LIDAR and the mounting of the sensors will be described in detail in section 14.6.

13.1 Adaptive Field of View

Since the LIDAR gives measurements around the entire drone, a method was developed to adaptively choose a sector of interest and filter out points that was not within this sector. This is to reduce the impact of differences in geometry along the surface. This is a reflection of the fact that the section of the surface straight in front of the drone, is the most important information for the contact operation.

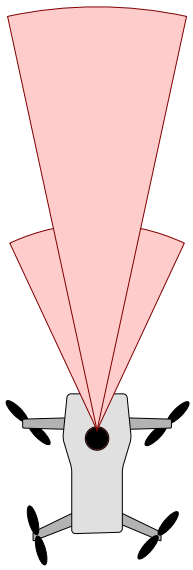


Figure 13.1: Adaptive sector

First, faulty measurements are filtered away by discarding elements outside the valid range specified by the sensor. Secondly, a sector is adaptively chosen based on the distance from the surface. Given the target width of the inspection surface w_t and the previous range r_p, the sector field of view angle α is chosen as

$$\alpha = 2\,\mathrm{atan2}\left(\frac{w_t}{2}, r_p\right) \tag{13.1}$$

In addition, each sector α is limited by $\alpha_{\min}$ and $\alpha_{\max}$.

Hence, the sector chosen tries to cover the same target width of the surface in front. When the drone is close to the surface, this will result in a wider angular field of view being chosen, while the angular field of view will be smaller when the drone is further from the inspection surface. An illustration of this is seen in fig. 13.1. Since the exact distance to the target surface may not be available, the range solution of the previous iteration is used.

All points outside the sector are discarded before the information is passed along to the algorithms developed in the following sections.

13.2 Regression

This section will describe an algorithm based on regression used to find the relative heading with respect to the inspection surface. Points remaining after the sector selection are used in a linear regression model [30] to find the vector parallel to the inspection surface and the distance from the surface to the Unmanned Aerial Vehicle (UAV). Given the scan points, the following matrices can be defined:

$$\mathbf{Y} = \begin{bmatrix} 1 & y_1 \\ 1 & y_2 \\ \vdots & \vdots \\ 1 & y_n \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} \tag{13.2}$$

where (x_i, y_i) is the coordinate of the i'th scan point in the body frame (x-axis forward, y-axis right) and β_1 and β_2 are the regression solution. β_1 and β_2 can be interpreted as the line's x-axis crossing (range) and slope respectively. Then the solution to the regression problem $\mathbf{Y}\boldsymbol{\beta} = \mathbf{X}$ is given as [30]:

$$\boldsymbol{\beta} = \mathbf{Y}^{+}\mathbf{X} \tag{13.3}$$

where $\{\cdot\}^{+}$ is the Moore-Penrose psudoinverse. The relative heading can be calculated from the line slope as

$$\psi_{\text{rel}} = \text{atan2}\left(\beta_1, 1\right) \tag{13.4}$$

Based on this model, data points that are outside a certain error threshold are filtered out. To minimize the effect of obscuring obstacles, such as the measurement probe, a second regression is performed on the remaining points to form the solution. This process is visualized in fig. 13.2. An extreme outlier can be seen in the point cloud in front of the drone, which affects the first regression solution (orange). However, the extreme outlier is outside the error threshold, and is thus filtered out. A second regression solution (blue) is then fitted, which corresponds better to the expected result. The error threshold was set at 0.05 cm, based on experiments conducted. In addition, no solution will be reported if the number of points after the first regression is too low.

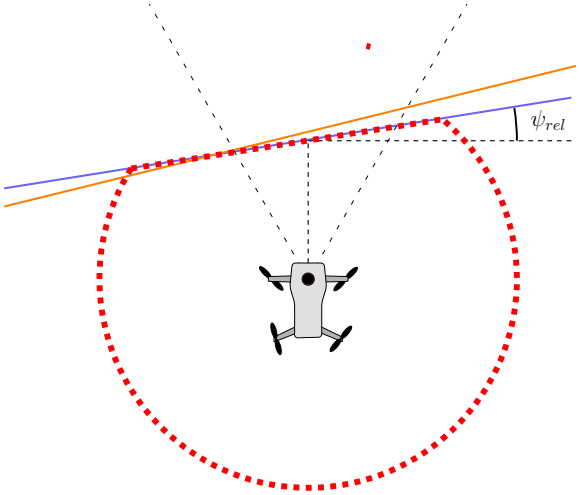


Figure 13.2: The effect of running two regressions. The first regression is shown in orange and the second regression is shown in blue

13.3 RANSAC

RANSAC is described in section 11.2 and the following will describe the model, parameter estimation and distance metric used. The line model is given as:

$$x = \beta_2 y + \beta_1 \tag{13.5}$$

Given two scan points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ in the body frame (x-axis forward, y-axis right), the line parameters can be estimated as

$$\beta_2 = \frac{x_j - x_i}{y_j - y_i}, \qquad \beta_1 = x_i - \beta_2 y_i; \tag{13.6}$$

Given another external point $p_e = (x_e, y_e)$, the distance to the line is given as

$$d = \frac{|\beta_2 y_e - x_e + \beta_1|}{\sqrt{\beta_2^2 + 1}} \tag{13.7}$$

The case when $y_j - y_i = 0$ (vertical line) is handled separately. The number of iterations k must also be chosen. Investigation of the laser scan points from the LIDAR showed that the inlier ratio varied between 95% in the best case to 45% in the worst cases. Based on eq. (11.2) it is possible to find the number of iterations needed to find a good solution with some probability. Setting the desired probability $p = 0.9999$, the inlier ratio $w = 0.45$ and the model order $n = 2$ in eq. (11.2) yields:

$$k = \frac{\log(1 - 0.9999)}{\log(1 - 0.45^2)} \approx 40.7 \tag{13.8}$$

Experiments showed that any k larger than this yielded good result, with linear increase in computation time. With $k = 50$ the computation time on the on-board embedded computer was approximately 0.003 s with some fluctuation due to the Linux scheduler (the embedded computer is described in section 14.2.2). The error threshold was set to 0.05 cm, based on initial experiments and geometrical considerations. In addition, no solution will be reported if the number of inliers are lower than 50% of the total number of points in the sector of interest.

In implementations of RANSAC it is recommended to do a regression on the inlier points after the algorithm has finished. This is because the solution model from the RANSAC algorithm is constructed from only two points (in the case of a line model). However, all inlier points are assumed to be generated by the same model, and doing regression on the inlier points gives a solution that is more robust to noise. Hence, this was also included in the implementation of this method.

13.4 Results of Experimental Verification

Different test scenarios were carried out to evaluate and verify the capabilities of the implemented solutions:

- Controlled Environment (section 13.4.1): Presents and compares the results of both algorithms in a controlled test setup. Sensor was not mounted on the drone platform.
- Circular Room (section 13.4.2): Discusses the performance of the methods in circular rooms and evaluate them using simulated data.
- Flight Test (section 13.4.3): Presents and discusses results from flights conducted in the test room described in section 18.1.1.

13.4.1 Controlled Environment

The test setup is shown in fig. 13.3. A printed protractor and a table was used to simulate different relative headings. The protractor and the LIDAR was aligned using floor markings.

(a) Overview (b) Protractor

Figure 13.3: Test setup for controlled environment

Test I: Close Proximity

The LIDAR was placed 0.45 m from the surface, with the sensors starting out perpendicular to the surface. The surface was then turned according to fig. 13.4. The dashed black lines marks the relative heading measured by the protractor, and it should be noted that the blue and orange line are difficult to separate as they are almost identical. Both methods showed excellent tracking of the surface, and a snapshot of the laser scan with the solution of the regression algorithm at $t = 56.14\,\text{s}$ can be seen in fig. 13.5.

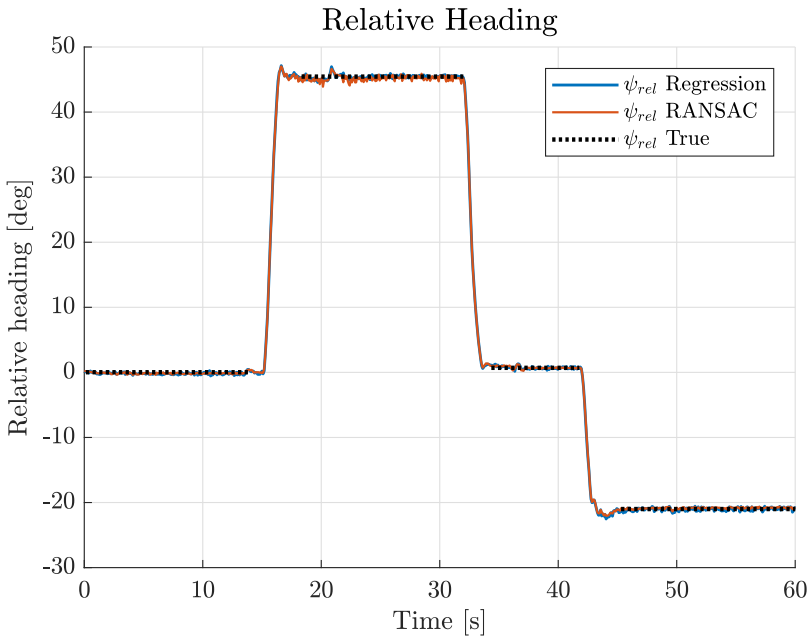


Figure 13.4: Relative heading to test surface

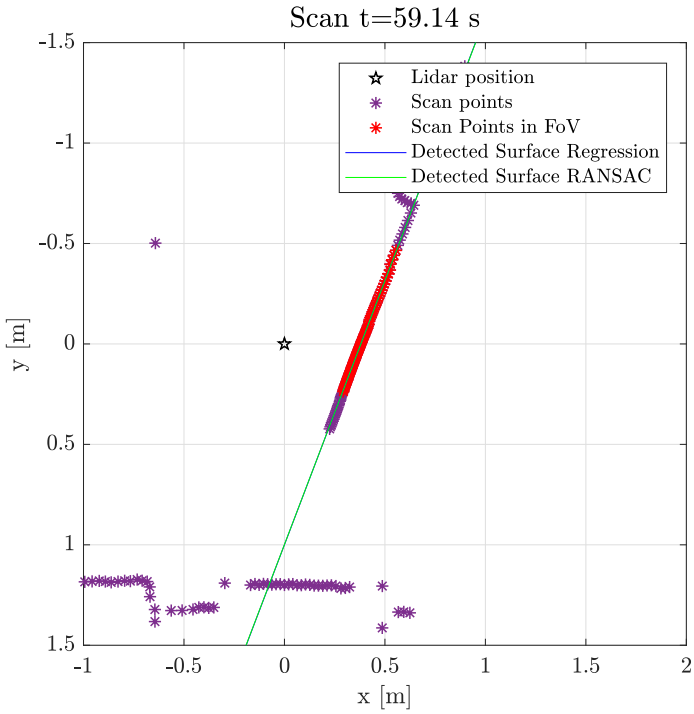


Figure 13.5: Scan image at $t = 56.14\,\text{s}$. Front of LIDAR is facing towards positive x

Test II: Obstruction

A test was also conducted with an object in front of the LIDAR to simulate the effect of the probe obstructing a small sector in front of the LIDAR. The obstruction can be seen in fig. 13.7, approximately 0.25 cm along the x-axis from the LIDAR position. It is essential that the methods are able to perform with these limitations, as the probe mounting will cause a similar blockage of the LIDAR sensor.

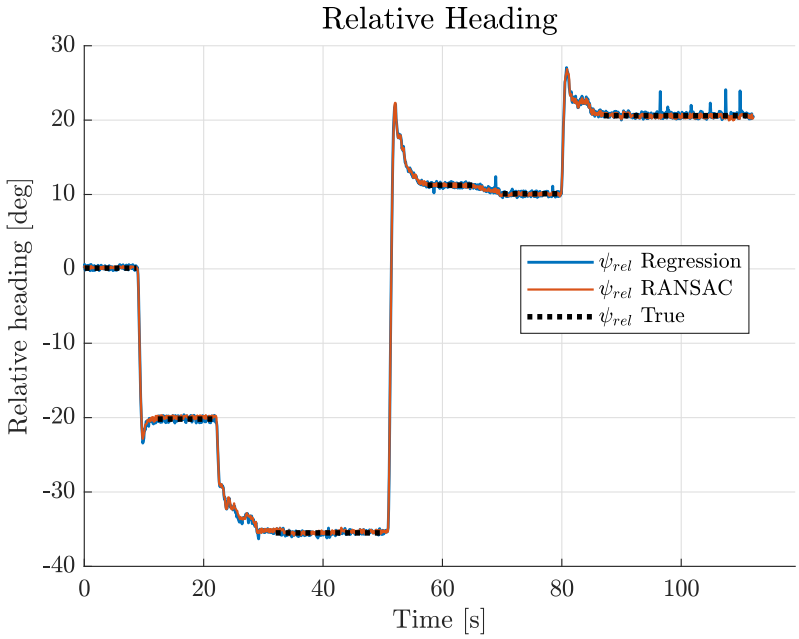


Figure 13.6: Relative heading to test surface

Once again the surface was rotated to simulate different relative headings as seen in fig. 13.6. The dashed lines displays the measurements from the protractor. Once again both methods are generally able to perform as expected; however, some differences between the methods become apparent. This is particularly visible from $t = 90\,\text{s}$ to $t = 110\,\text{s}$, where spikes of about 5° can be seen in the regression estimate. These spikes are also visible at other time instances of the experiment.

This was similar to other experiments conducted, where it was apparent that the RANSAC method is able perform better at lower inlier ratios compared to the regression method. This is linked with the fundamental assumptions of the two methods, as regression assumes that all points are generated by the same model, as opposed to RANSAC where the outliers comes from a different generative distribution. This will be discussed further in section 13.5.

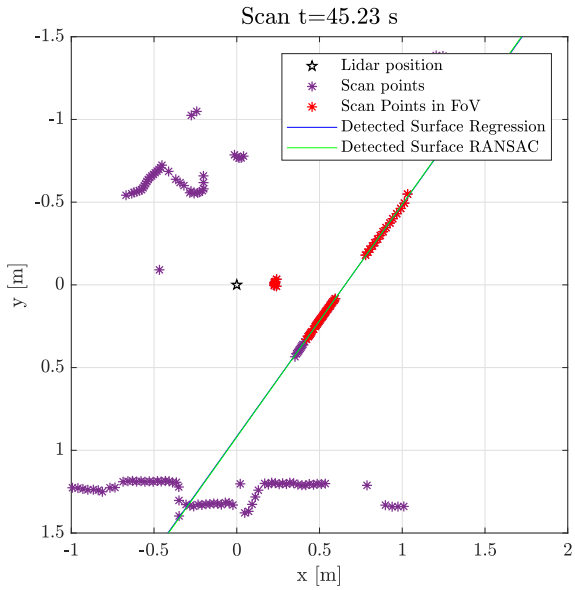


Figure 13.7: Scan image at $t = 45.23\,\text{s}$. Front of LIDAR is facing towards positive x

Variance comparison

To better quantify the quality of the solution, an experiment was also carried out to determine the variance of each of the methods. The relative heading was adjusted from $-30°$ to $30°$ with steps of $5°$. No obstruction was present. The variance was found at each step, and the experiment was carried out both close to the surface (distance of $0.45\,\text{m}$) and further away (distance of $2.65\,\text{m}$). The results are shown in fig. 13.8.

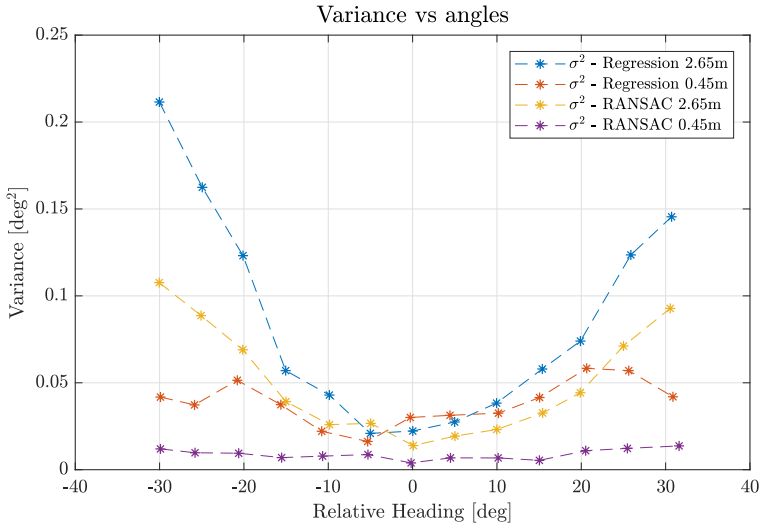


Figure 13.8: Variance plotted against heading

The overall results show that the variance is small for all scenarios; however, there are differences correlated with both distance and method. The RANSAC method perform generally better with respect to variance. On short distances the RANSAC method has very small variance (Purple line in fig. 13.8), and shows no clear correlation with the magnitude of the angle. From the results, it is apparent that the variance increases with the distance to the surface (blue and yellow line are furthest away). This is expected, as the noise on each individual scan point increases with the distance from the sensor. From fig. 13.8 in can also be seen that the RANSAC method performs similar on the longer distance as the regression method on the short distance. However, the regression method performs significantly worse on the longer distance, with variance also being clearly linked to the magnitude of the angle. This shows that the RANSAC method has the most robust performance with noisy measurements.

13.4.2 Circular Room

This section will briefly discuss some of the problems related to the definition of relative heading in circular rooms, and evaluate the performance of the two methods in these conditions.

The definition of relative heading with respect to a circular room is clearly not well defined. However, it is still possible to define the desired behavior in this situation. In the contact based inspection scenario, the relative heading is used to guide the drone to an orientation where it is facing perpendicular to the surface. A more general definition of this configuration, is that the drone is facing in the direction of the shortest distance to the inspection surface. Hence, a reasonable behavior for the heading estimation in the case of a circular room is to give the relative heading with respect to the tangent line at the closest point on the inspection surface. In fig. 13.9 this corresponds to the situation when the drone is orientated directly along the x-axis in the global view.

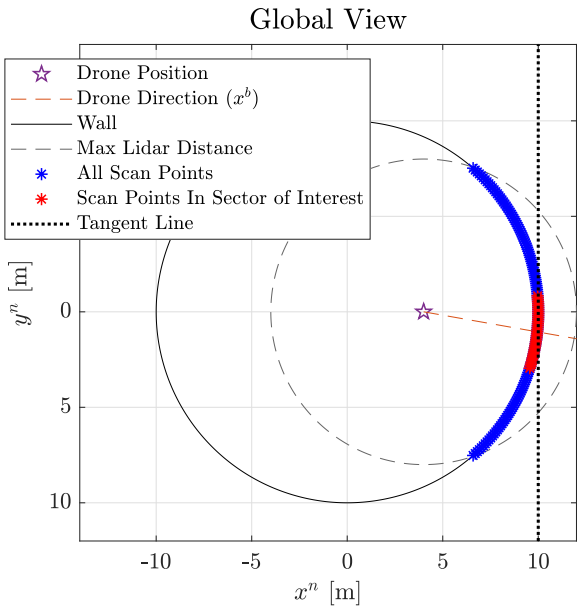


Figure 13.9: Global view of the drone position and environment

However, on the simulated data, it is apparent that both methods will slightly underestimate the heading relative to the tangent line. This can be seen in fig. 13.10, where the estimated heading is about 6°, while the true relative heading with respect to the tangent line is 7.2°. It should be noted that the tangent line in fig. 13.10 has been translated center of the laser scan, to better illustrate the heading difference.

However, the underestimation of the heading is not a problem if relative heading is used as the control variable, with the goal of driving it to zero. While the feedback variable is slightly lower than reality, the controller will still drive it towards 0°, giving the desired behavior of aligning the drone to the closest surface. This is similar to controllers based on linearized models driving a processes towards the operating point used in the linearization process.

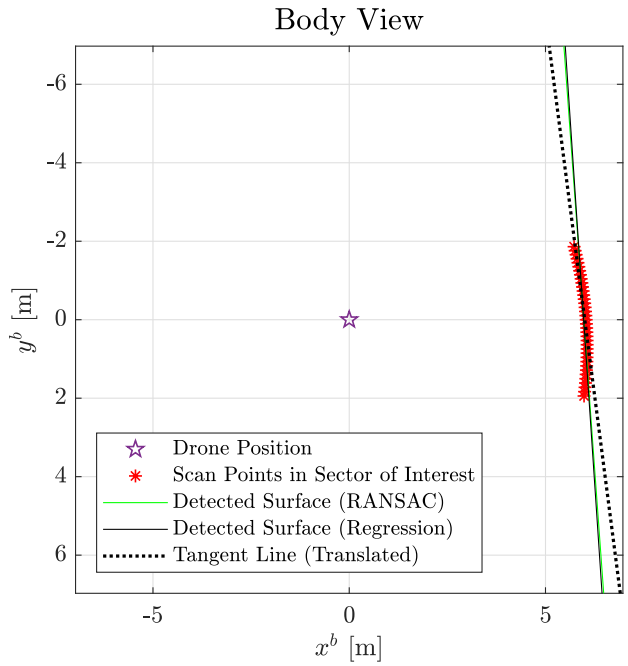


Figure 13.10: Body view. Forward direction along the x_b- axis. Only showing LIDAR scan points inside the adaptive sector

For the regression method a guarantee can be made under the assumption of idealistic data, that the method will only give 0° when looking straight at the tangent line at the closest point on the circular surface. This is because this is the only time the algorithm will see the scan points as symmetric around the body x-axis, which results in the regression algorithm giving a heading of 0°.

For the RANSAC method it is more difficult to make the same arguments and guarantees about the behavior of the algorithm, since it is based on probabilistic methods. However, the most optimal model will be the same as the model found by the regression algorithm. The problem is that during the k iterations of RANSAC, there is no guarantee that this solution is found. Nevertheless, the RANSAC method shows equally good performance in the simulated scenario in this section. It also shows the same behavior of underestimating the heading as the regression method.

13.4.3 Flight Test

A manual flight test was also conducted in the test room described in section 18.1.1. In this test, the algorithm was configured with a sector of interest in all four cardinal directions of the body frame. This is because it was also desirable to find the range to all surfaces, in order to track the relative placement of the drone in the environment. According to the definition in section 11.1 this gives a local position estimate, which can be used for control purposes. The implementation of this will be discussed in section 16.3.

The result can be seen by following the video in the QR-code in fig. 13.11. A snapshot of the result is also showed in fig. 13.13. As the previous sections revealed that the RANSAC algorithm gave the most robust results, only the results of this method is shown in this video, represented by the blue lines. The regression method was also tested using the same dataset; however, it showed far worse performance, and thus it was not included in this report. The reader is highly recommended to watch the video to fully understand the interpretation of the results.

The room is cluttered with desks and a net to protect to operator in the backward direction, which causes loss of tracking at some instances during the flight. Since the inlier ratio for an accepted solution was set high, this is to be expected.

In the forward direction two separate surfaces are present. From $t = 38\,\mathrm{s}$ to $t = 48\,\mathrm{s}$ in the video it can be seen that the algorithm does not produce any answer when the surface in front of the drone switches from the metal plates in the front to the wall in the back. This is desired behavior, as ambiguities should not produce a valid result. For the same reason, tracking is lost in the left direction when the drone is close to the surface ($t = 25\,\mathrm{s}$), as the pilot struggles to control the orientation of the drone.

Figure 13.11: QR-code for manual flight test (https://youtu.be/RvXeqMnb9z8)

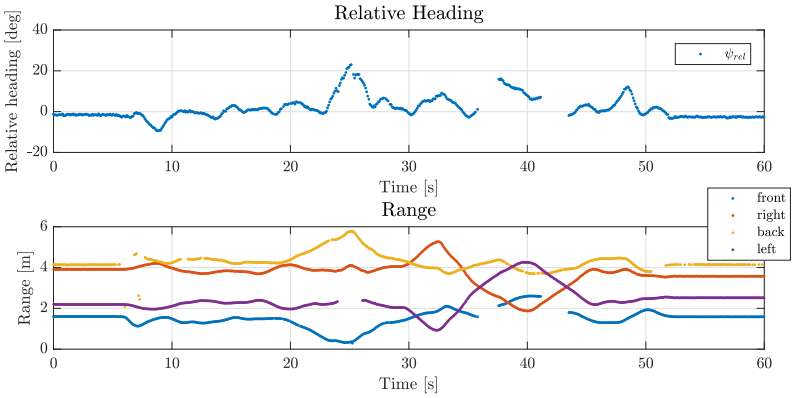


Figure 13.12: Relative heading and ranges during the manual test flight

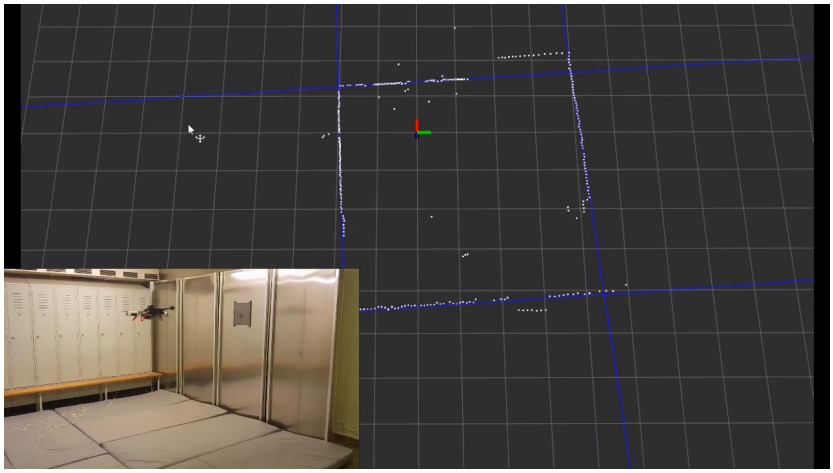


Figure 13.13: Screenshot of the RANSAC algorithm running in all four sectors

13.5 Conclusion

The most noteworthy drawback of regression comes from the linear model used. The model implicitly assumes that the points in the laser scan comes from a planar surface, which might not be the case in the presence of obstacles or other distinct features in the surface. The adaptive adjustment of the field of view angle is an attempt to lower the impact of these issues, and helps select the points used to form the regression solution from the most relevant regions of the scan. However, if the violation of the planar assumptions is to extreme, the performance of the regression will be poor.

Generally, the RANSAC method are able to handle a much higher outlier ratio then the regression method. This comes from the fundamental underlying assumption of the two methods: Regression assumes all data points are generated by some model (with noise), while the RANSAC method assumes that the data points can be classified as either inliers or outliers to a certain model. The latter assumption clearly fits the data coming from the LIDAR better. Hence, both methods provide robustness against sensor noise, while the RANSAC model gives better rejection against surface geometry and obscuring obstacles, despite the attempt to filter outliers by doing a double regression.

The behavior in circular rooms is also satisfactory for both algorithms, under the assumption that the control objective is to regulate the heading to 0°. This will suppress the slight underestimation of the heading that was described in section 13.4.2. It is also possible to extend the RANSAC implementation to fit a circular model instead; however, this is left as future work.

Pitch and roll angles will also affect the performance of the algorithms described in this part. For the relative heading estimate based on a 2D scanning LIDAR, the assumption is that the roll and pitch angle will be small. If this is not the case, it is difficult to interpret the result of the algorithm in any meaningful way. Since the contact based inspection scenarios will be performed with low horizontal velocities and hence small displacements in roll and pitch, this was concluded to be a reasonable limitation.

The solution technique described in the above chapters also has a potential for further extension:

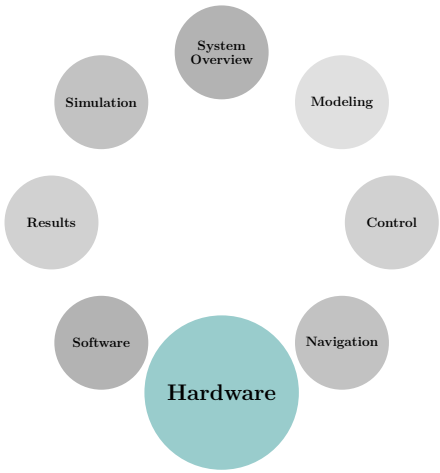
Assumption about the geometry of the surrounding environment can be done, where a flat surface assumption is the most obvious. Thus, the orthogonal distance to the surfaces in all sectors can be used instead. This would give a global position estimate relative to a coordinate system attached to the room, under the assumption that it is rectangular. However, in an attempt to make the solution as general as possible with respect to the geometry of the room, this has not been done in this thesis.

The lines extracted in the regression algorithm can also be used as features in a SLAM algorithm, such as in [68]. As the number of features are small, this can be done in a SLAM algorithm based on an extended Kalman Filter [16]. It is also possible to use the entire scan in a SLAM algorithm, such as in [46]. However, these techniques are advanced and considered beyond the scope of this thesis.

It should also be noted that the solution based on the 2D scanning LIDAR potentially introduces a time delay into the system. The sensor used in this thesis dispatches a scan of 360° every 0.1 s, which means that some parts of the laser scan potentially is more than 0.1 s old. If a Kalman Filter is used to filter these measurements it should be formulated on error form in order to give better rejection against these time delays [63].

Part V

Hardware



Chapter 14

Drone Platform

This chapter will provide details about the hardware of the drone used in this thesis. First, a brief overview will be presented, before the main components of the drone are described in the order below.

1. **Frame and Motors** (section 14.1)
    - Motor Controllers (section 14.1.1)
2. **Flight controller and Computer Module** (section 14.2)
    - Pixhawk CUBE (section 14.2.1)
    - Toradex iMX6 Colibri (section 14.2.2)
    - Motherboard (section 14.2.3)
3. **Range Sensor** (section 14.3)
4. **IMU** (section 14.4)
5. **Ultrasonic Probe** (section 14.5)
6. **LIDAR** (section 14.6)

The design of the platform, as well as all hardware components, have been supplied by Scout DI. During this thesis, the ultrasonic probe and 2D scanning LIDAR has been integrated onto the platform, as well as general maintenance and assembly of the rest of the platform. Section 14.7 will briefly present another hardware platform used for additional testing.

First, an exploded view of the drone and all its parts are shown, followed by block diagram showing the connections and communication protocols for the different hardware components. Finally, a picture of the fully assembled drone with all mission payloads is shown. The intent of the following is to provide the reader with an overview of the capabilities of the platform used to conduct the experiments in this thesis. Hence, it is kept intentionally brief to not overwhelm the reader with the implementation details.

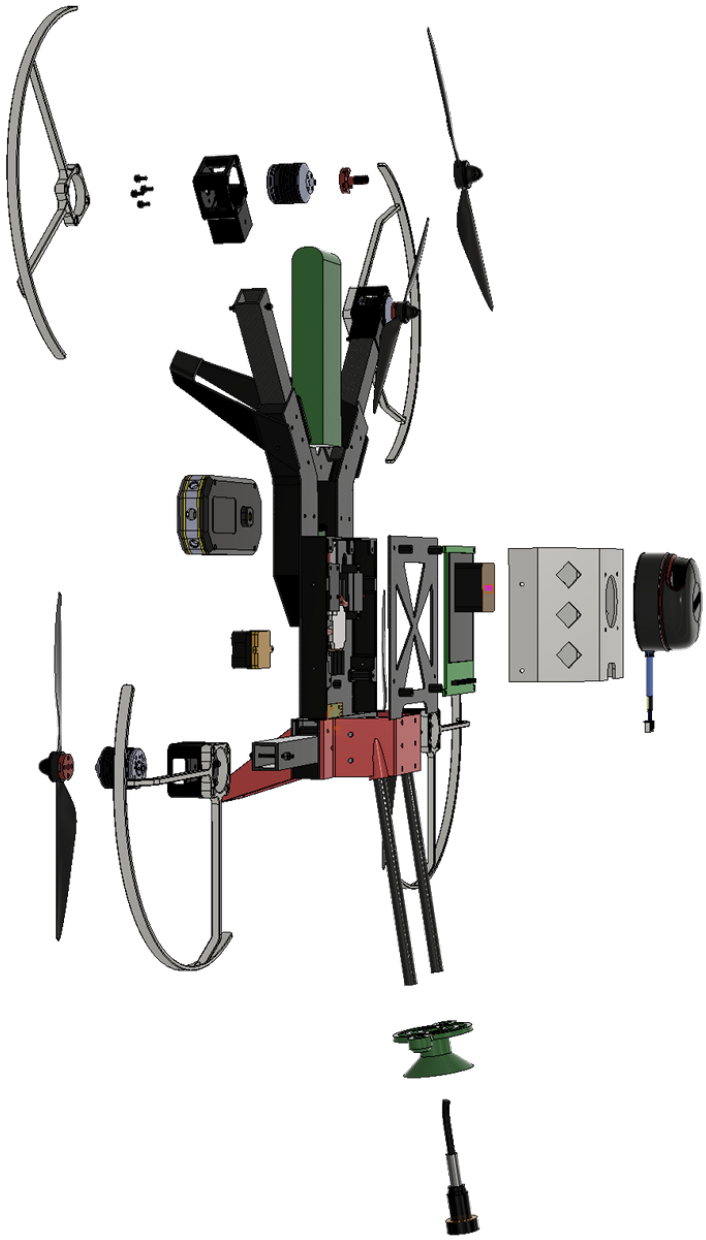


Figure 14.1: Exploded view of the drone and all hardware components. (Courtesy of Scout DI)

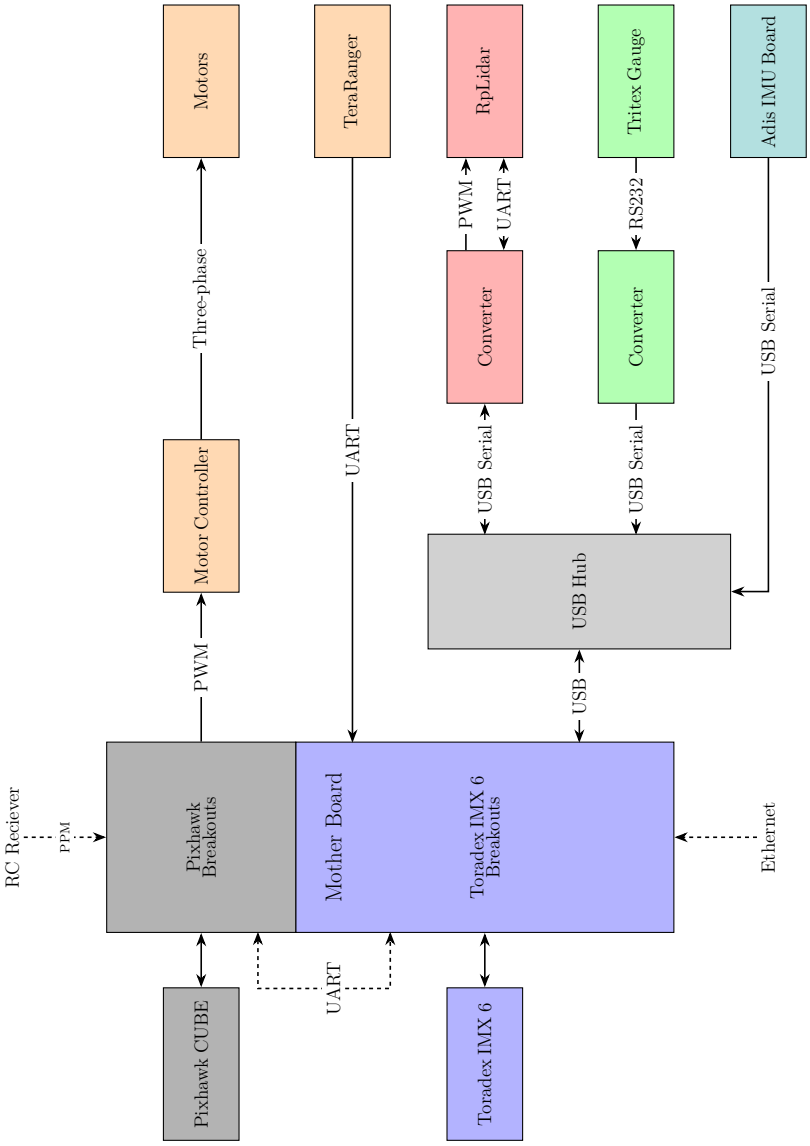


Figure 14.2: Hardware connection and communication protocols

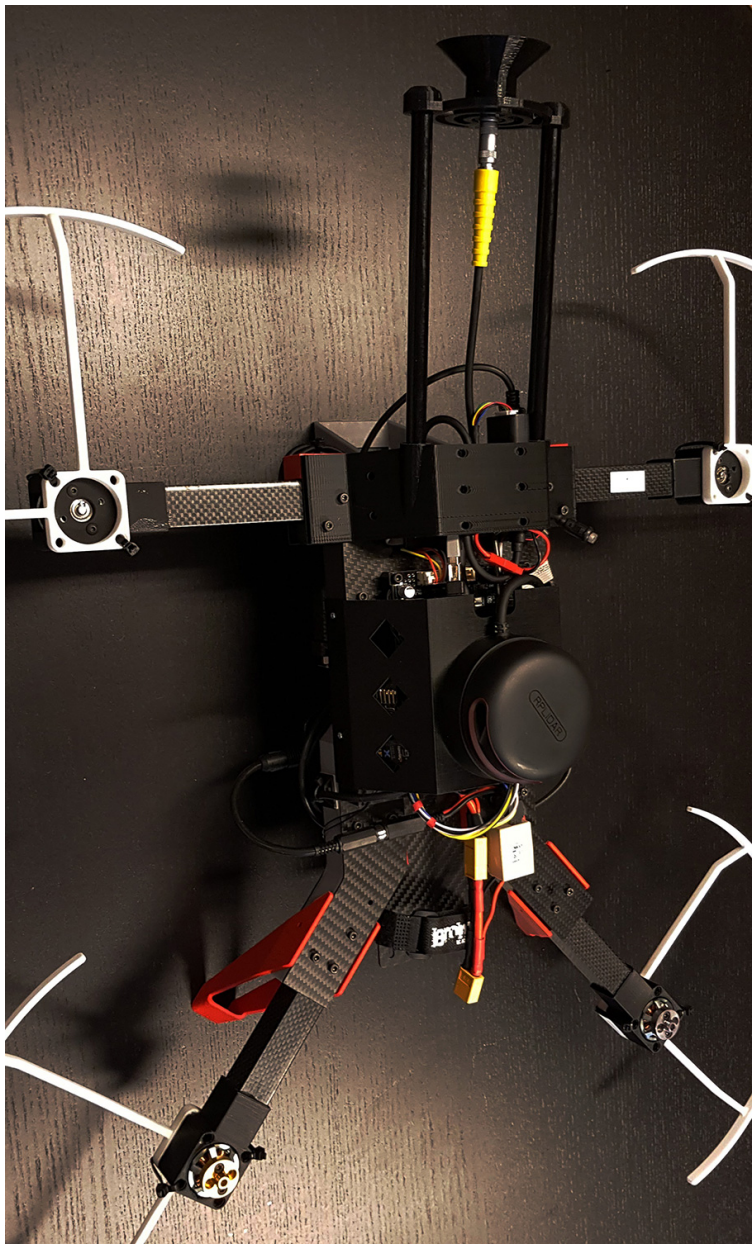


Figure 14.3: Fully assembled drone with all mission payloads

14.1 Frame, Motors and Power Supply

The drone frame consist mostly of carbon fiber plates in conjunction with 3D printed plastic, such as for the 3D printed casings for the motors and the landing gear.

The motors are brushless AC motors of the type KDE2315XF-2050 manufactured by KDE Direct[1]. This motor is more powerful than what is commonly used with drones of this size. This comes at the expense of a higher power consumption, resulting in shorter flight time. The reason for this design choice was that during initial tests with less powerful motors, the drone would experience temporary falls in quick maneuvers. This was due to some of the motors experiencing saturation in thrust potential, meaning that attitude could only be corrected by lowering the thrust of all motors, and thus resulting in loss of altitude. Since this drone platform is intended for development, the choice was made to have excessive thrust potential rather than do premature optimization on the combination of weight, motors and propellers.

The propellers are 9.5 inch in diameter with two blades at a pitch of 45°.

The drone is supplied power from a 3S LiPo battery, which has a nominal voltage of 12 V. In addition, the drone is equipped with a BEC (Battery Eliminator Circuit)[2] to convert the input voltage from the main battery to different voltages levels, and a Power Line (PL)[3] sensor to measure the voltage and current consumption.

[1] https://www.kdedirect.com/products/kde2315xf-2050
[2] https://www.mauch-electronic.com/apps/webstore/products/show/7588500
[3] https://www.mauch-electronic.com/apps/webstore/products/show/7594481

14.1.1 Motor Controllers

The drone is equipped with a T-Motor F55A Pro[4] Electronic Speed Control (ESC). The ESC controls all four motors, and is capable of delivering 75 A burst and 55 A sustained current. Inputs to ESC comes from the Pixhawk CUBE described in section 14.2.1.

The full integration of the motors, ESC and power modules can be seen in fig. 14.4.

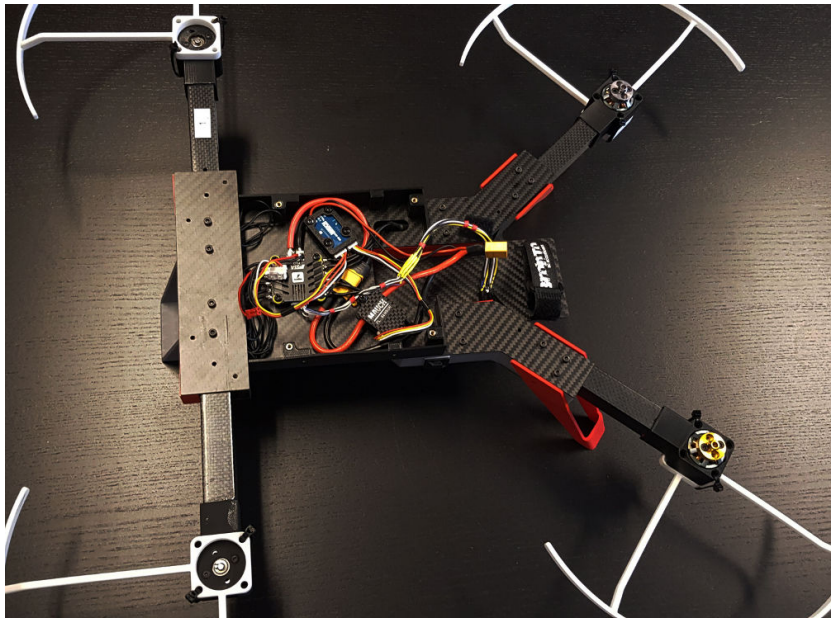
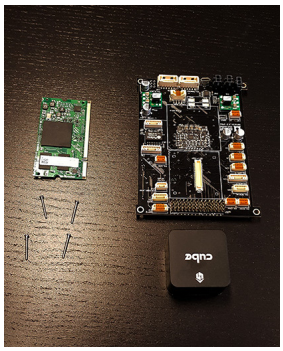


Figure 14.4: The drone frame including motors, motor controller and RC-receiver

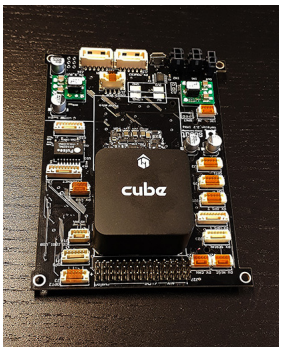
[4] https://www.getfpv.com/t-motor-f55a-pro-55a-3-6s-blheli32-4-in-1-esc.html

14.2 Flight Controller and Computer Module

This section will describe the specification of the computational hardware used on the drone platform. First, the flight controller is presented, followed by the main computer module. The last section shows the motherboard, which features breakouts for both of the aforementioned computational modules.

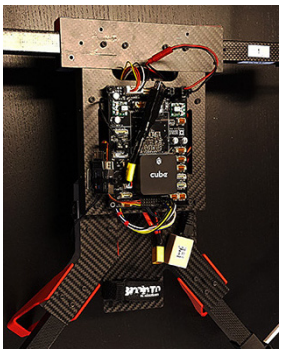


(a) All components



(b) Motherboard (front)

(c) Motherboard (back)



(d) Assembled on drone

Figure 14.5: The Motherboard (designed and developed by Scout DI), Pixhawk CUBE and Toradex iMX6 Colibri

14.2.1 Pixhawk CUBE

The low-level autopilot running on the drone is the Pixhawk CUBE[5] hardware with Ardupilot[6] software stack. The hardware platform is popular and adopted for both recreational and professional purposes. Ardupilot includes software specific for both multirotors and fixed wings, called ArduCopter and ArduPlane respectively. Ardupilot is open-source and alterations to the code has been made by Scout DI to enable indoor flight without compass aiding, and control of the desired attitude of the multirotor using outputs from an on-board computer.

In addition to the attitude control, the most important features of the Ardupilot is that it also provides attitude estimates and relays Remote Control (RC) receiver inputs to the on-board computer. The RC-receiver communicates with the Pixhawk using Pulse Position Modulation (PPM).

14.2.2 Toradex iMX6 Colibri

The Toradex iMX6 Colibri is a Computer on Module (COM), which is a subtype of an embedded computer system. This module runs all of the frameworks and high-level control software on the drone. The iMX6 is capable of running a range of different operating systems, but in this case it runs a Linux distribution which will be described in section 15.5.

The Toradex iMX6 is powered by a Cortex A9 dual core CPU at 1 GHz, and features interfaces such as Inter-Integrated Circuit (I2C), Universal Serial Bus (USB), and Universal Asynchronous Receiver-Transmitter (UART). The full specifications for the Toradex iMX6 Colibri can be found online[7]. The Toradex iMX6 can bee seen in fig. 14.5a and is mounted on the backside of the motherboard, as seen in fig. 14.5c.

14.2.3 Motherboard

The motherboard is designed and developed by Scout DI, and is the glue of all the hardware components on the platform. It provides power at different voltages and connects the main on-board computer and the low-level Pixhawk together over a UART interface. In addition, the motherboard also allows for easier integration of different sensors by exposing interfaces for common hardware

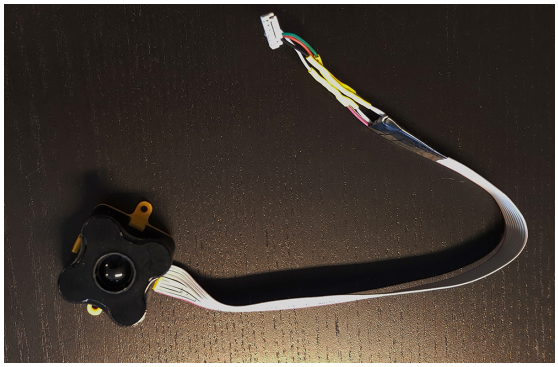
[5] https://docs.px4.io/en/flight_controller/pixhawk-2.html
[6] http://ardupilot.org/ardupilot/
[7] https://www.toradex.com/computer-on-modules/colibri-arm-family/nxp-freescale-imx6

communication protocols such as I2C, UART and USB from the iMX6. The motherboard and the integration with the iMX6 and Pixhawk can be seen in fig. 14.5.

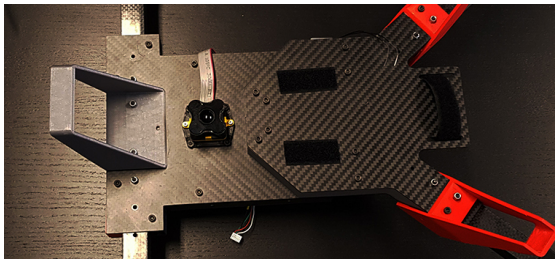
No wireless modules are installed on this development drone, and thus a Ethernet cable was connected to the drone during flight to ensure communication with control applications and transmission of debug data to the ground station.

14.3 Range Sensor

The range sensor used in this thesis is a TeraRanger Evo[8] 3m produced by TeraBee, as seen in fig. 14.6a. It is a ToF sensor, and uses IR light to measure the range.



(a) TeraRanger Evo 3m with custom UART cable



(b) TeraRanger mounting

Figure 14.6: TeraRanger

[8] https://www.terabee.com/shop/lidar-tof-range-finders/teraranger-evo-3m/

The TeraRanger features a range of backboards to expose different connection interfaces, such as USB 2.0 Micro-B, UART and I2C. In this case the UART interface was chosen, and a custom cable was soldered to connect to the serial interface of the iMX6 through the motherboard. The sensor has a field of view of 2°, and produces range measurement within $0.1 - 3\,\text{m}$ at an update rate of 100 Hz.

The TeraRanger was mounted in a downward facing direction to measure the altitude of the drone, as shown in fig. 14.6b.

14.4 IMU

A high grade Inertial Measurement Unit (IMU) was also mounted on the drone. This was done using a circuit board designed and developed by Scout DI, and features an ADIS 16465[9] IMU and a Teensy[10] microcontroller to handle the low-level communication and timing with the IMU. The Teensy microcontroller communicates with the IMU over Serial Peripheral Interface (SPI) and forwards messages to the iMX6 over USB.

The board can be seen in fig. 14.7, and was mounted below the Pixhawk in order to keep it as close to the center of mass as possible.

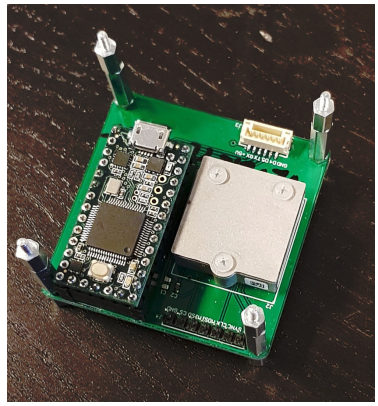


Figure 14.7: The ADIS IMU and the carrier board with the Teensy microcontroller. Designed and developed by Scout DI

[9] https://www.analog.com/en/products/adis16465.html
[10] https://www.pjrc.com/store/teensy32.html

14.5 Ultrasonic Probe

This section will give a more thorough introduction to the ultrasonic probe used to measure the metal thickness of inspection surfaces in this thesis. Since the reader is assumed not to be familiar with the ultrasound technology, a brief introduction will be given in the next section. Thereafter, specifications of the Tritex Gauge[11] will be presented, along with the modifications needed to mount the probe system onto the drone platform.

14.5.1 Background

The principle behind ultrasonic thickness measurements are similar to any time of flight technique. Ultrasonic sound waves are transmitted into a material and reflected back to the sender when they hit areas with different densities. If the speed of sound in the material is known, the thickness can be calculated using the measured time between sending and reception of the sound waves.

However, there exist a couple of challenges to get reliable measurements. Since air have significantly different properties from most other solid materials, it is ideal to have complete contact between the probe and the target material. This is normally done using a special gel that ensures contact, much like examinations for pregnant women. However, there is still a difference in the properties of the gel and the target material that causes reflection of the sound waves.

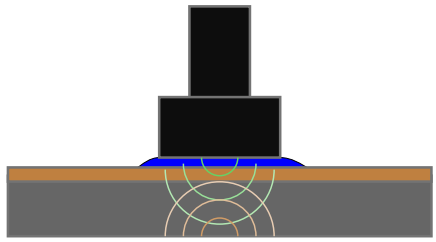


Figure 14.8: Illustration of ultrasound principle for thickness measurement

The sensor uses a technique called multiple echo to ignore the thickness of the coating during measurements, as illustrated in fig. 14.8. In simplistic terms this means that the reflections of all the different materials are measured, but time traveled in other materials than the target are subtracted away. Hence, it is possible to get valid readings even with the presence of coating and corrosion.

[11] https://www.tritexndt.com/product/multigauge-6000-drone-thickness-gauge

14.5.2 Integration

In order to mount the probe onto the drone, Scout DI 3D designed and printed a custom bracket and holder for the probe head. The holder in front was printed using a flexible plastic material in a gimbal-like structure, such that the holder becomes a flexible joint at the tip of the probe. This ensures sufficient mechanical compliance in the structure, while the plastic also gives a weight advantage compared to other methods based on metal springs. The mounting bracket and probe front can be seen in the top left part of fig. 14.9.

All parts necessary to mount and power the ultrasonic probe can be seen in fig. 14.9. Enumerated from left to right starting at the top row, the parts are:

1. Custom 3D printed mounting for probe head with flexible joint and carbon fiber tubes
2. 3D printed bracket for attaching the probe
3. Serial to USB converter
4. Custom created power supply cable
5. Probe head
6. Probe cable
7. Probe electronics box

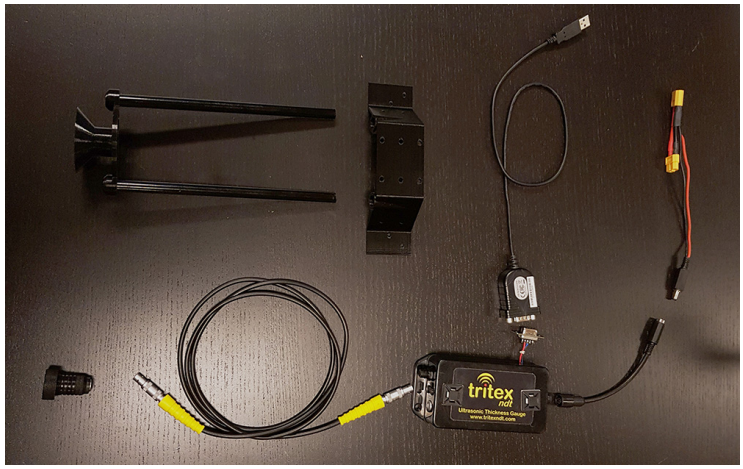
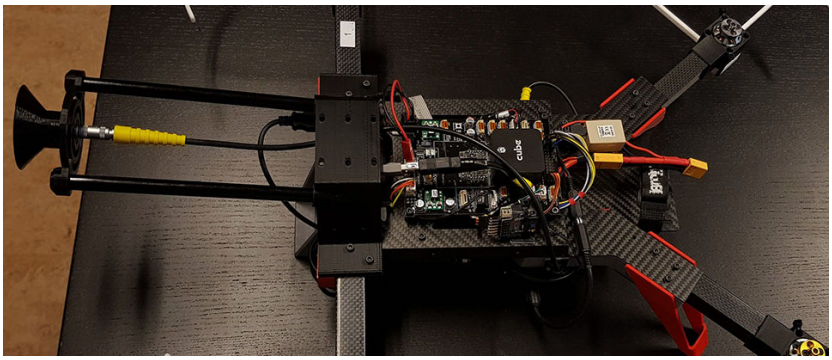
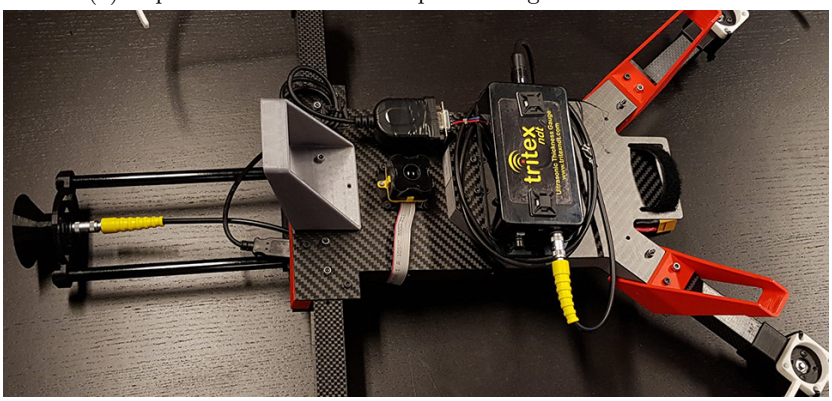


Figure 14.9: All parts of the ultrasonic probe and mounting brackets

The serial interface was exposed by soldering an RS232 cable to the internal breadboard of the ultrasonic hardware. Furthermore, an RS232 to USB converter was used to connect to the motherboard, since the motherboard does not feature circuitry to handle the differential voltage of the RS232 interface. The packets coming from the probe hardware containing the measurements are then interpreted by the on-board software. This is done using a custom driver, which will be further described in section 16.2.2. The finished mounting of the ultrasonic probe can be seen in fig. 14.10.



(a) Top view of the ultrasonic probe integrated on the drone



(b) Bottom view of the ultrasonic probe integrated on drone

Figure 14.10: Overview of the integration of the ultrasonic the probe

14.6 LIDAR

The LIDAR used is an RPLIDAR A2[12] developed by SLAMTEC. It is a 2D scanning LIDAR, powered by a motor that causes the upper part of the LIDAR housing to rotate.

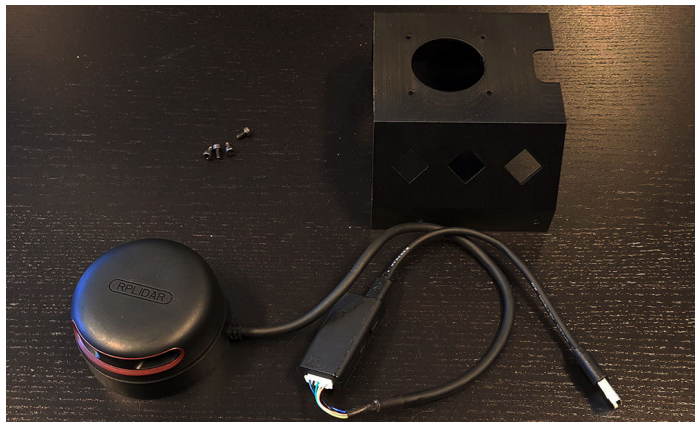
According to the specifications, the LIDAR has a measurement range of $0.15 - 12\,\text{m}$, with an angular resolution of 1°. The full 360° laser scan is provided at a frequency of 10 Hz.

To mount the LIDAR on the drone, Scout DI 3D designed and printed a custom bracket, as seen in fig. 14.11. The drone was connected to the motherboard using a USB Hub, and the fully integrated LIDAR can be seen in fig. 14.3.

A custom sensor driver was created in order to communicate with the hardware components of the LIDAR, using an SDK provided by SLAMTEC. This is described in detail in section 16.2.1.

One issue with this specific LIDAR model became apparent when the LIDAR was used at the Falck Nutec facilties (described in section 18.1.2). The performance of the LIDAR in the tank was significantly worse compared to all other environments where the LIDAR was tested. Even though the max range is specified as 12 m, the LIDAR was unable to get readings of surfaces more than 4.5 m away. One possible source of the issue is that the range specifications are given for highly light emitting surfaces, such as a white wall. A darker colored surface will absorb more of the light emitted from the LIDAR, and thus reducing the maximum range. However, since the problem was not detectable when the drone was on ground, it is likely also correlated with the vibrations caused by the motors when the drone is airborne. It appears that these two error sources combined highly degrades the sensor performance. As an example, no range information was available in the lateral direction when the drone was facing the tank walls. Despite the limitations this poses on the navigation methods, several successful flights were carried out in the tank environment (see section 18.3).

[12] https://www.slamtec.com/en/Lidar/A2



(a) All parts for the LIDAR

(b) RPLIDAR on custom mounting

Figure 14.11: RPLIDAR integration

14.7 The Scout 135 Hardware Platform

During the work on this thesis Scout DI developed a new version of their inspection drone, called the Scout 135. This drone is closer to the intended production version, and all controllers and navigation solutions developed in this thesis will also be tested using this platform. This is done to show that the methods developed are hardware agnostic, but also provide insight into how the hardware design influence the performance of the controllers and interaction with the environment. The Scout 135 can be seen in fig. 14.12, and this section will describe the main differences between the two platforms. It should be noted that all design and hardware integration has been done by Scout DI, and only platform specific tuning of controllers has been done during this thesis. The results of flight tests with Scout 135 can be found in section 18.3.

The platform addresses many of the concerns of indoor flight by integrating the propellers into the chassis itself, and generally improving robustness in case of contact or crash. Since this drone is closer to the production model, more optimization has been done on the combination of weight, motors and propellers. The propellers are 6 inch in diameter with three blades at a pitch of 42°, and is powered by 4S LiPo batteries with a nominal voltage of 14.8 V.

With respect to the sensor and computational hardware, there are almost no differences between the two platforms, except that the downwards facing TeraRanger has been replaced with a Garmin LIDAR-Lite v3HP[13]. Other than having longer range, this sensor provides the exact same capabilities as the TeraRanger used on the development platform.

In addition, the platform also has a camera mounted on a gimbal, which allows for streaming of live video from the drone. The drone has also been fitted with Light Emitting Diods (LEDs) to provide some vision in poorly lit environments.

[13] https://buy.garmin.com/en-US/US/p/578152

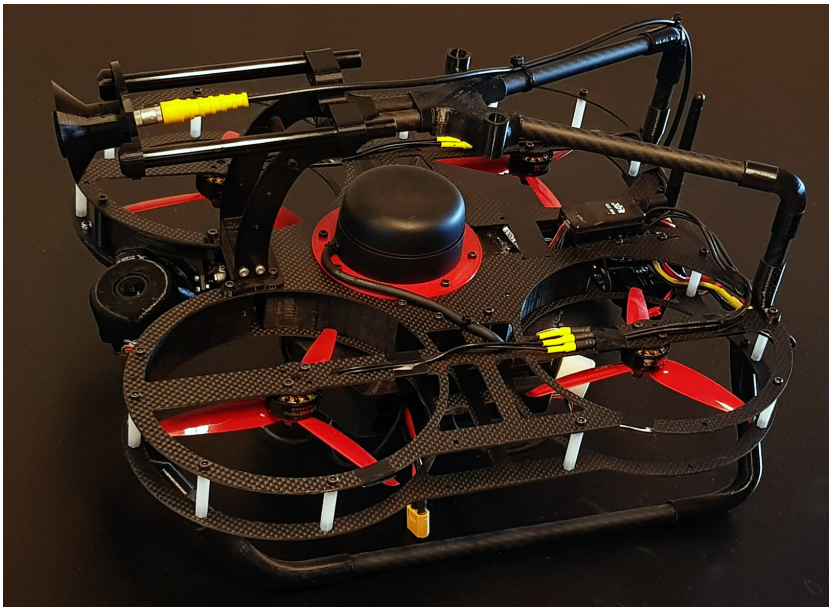
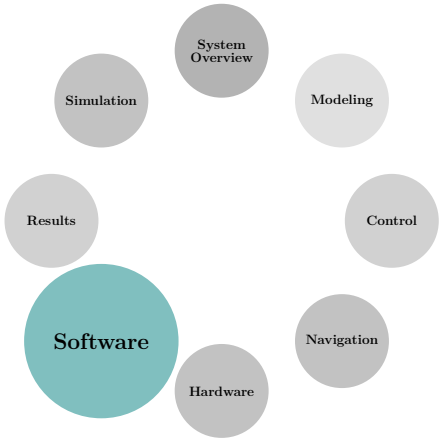


Figure 14.12: The Scout 135 platform

Part VI

Software



Chapter 15

Background

This chapter will provide a short introduction to the software framework used for the implementation and experimental validation of the controllers and navigation methods in this thesis. First, an overview of the LSTS Toolchain will be given, before the most relevant components are presented in more detail.

15.1 The LSTS Toolchain

The LSTS toolchain [57] is a software framework designed for operations with multiple, heterogeneous vehicles. The toolchain is developed by the Underwater Systems and Technology Laboratory (LSTS) at the University of Porto, Portugal. While the toolchain originally was designed for use with underwater vehicles, the software is now used by researchers and commercial companies to facilitate advanced operations with surface vessels, UAVs and underwater vehicles.

The LSTS toolchain consists of the following components. Note that only the items in bold will be covered, as these are the most relevant for this thesis.

- **DUNE: Software developing framework** (section 15.2)
- **IMC: Communication protocol** (section 15.3)
- **Neptus: Ground Control Station** (section 15.4)
- GLUED: A minimal Linux distribution
- Ripples: Communications hub for data (cloud service)
- ACCU: Android Command and Control Unit

In addition, a quick overview of the Linux distribution running on-board the drone will be presented in section 15.5.

15.2 DUNE

Unified Navigation Environment (DUNE) [58] is a software framework that provides a runtime environment for unmanned systems. It is written in C++ and runs on all major operating systems. The DUNE source code can be found online at github[1].

Task are the base concept in DUNE, and are effectively configurable submodules that run in parallel. A DUNE task can serve many different purposes, interfacing with hardware; translating higher-level goals to low-level actuator commands; and monitoring the state of the vehicle.

An important feature of DUNE is that all tasks are compiled together to form a single binary executable. This separates DUNE from other popular runtime frameworks such as the Robot Operating System (ROS) (see section A.2), where nodes or modules are run as separate executables. Compared to system level communication sockets, task communication using shared memory is much faster. The tasks communicate using a message protocol called Intermodule Communication (IMC), which will be detailed in section 15.3.

The benefit of the modular approach is the ease which a task may be replaced or improved. Seen from a system perspective, a new task need only provide the same information to the system as the ones it replaces. Hence, a task estimating position using beacons can easily be replaced with a task estimating position based on visual odometry, as long as they both broadcast the same position estimate messages. Since all broadcasted messages are available to all tasks, new tasks can easily access information it needs to perform its purpose. However, care should be taken when designing the complete system to ensure minimal information dependency for easier maintenance.

The DUNE framework provides core tasks that keeps track of the state of the vehicle, in addition to essential functionality such as logging and transmission of data to the ground station (Neptus, see section 15.4). DUNE also has support for the Micro Air Vehicle Link (MAVLink) protocol [60], which is a popular communication protocol for small unmanned vehicles. MAVLink is for

[1] https://github.com/LSTS/dune/

example used to communicate with low-level controllers, such as the Pixhawk (see section 14.2.1).

15.2.1 Run Configurations

Configuration files specify which tasks are executed, in addition to configuration parameters for each of these. Thus, different vehicles may use the same tasks, but with different parameters. Configuration files can include other configuration files, and form a tree-like structure. An illustration of the configuration file structure is shown in fig. 15.1. This allows a task to specify a set of default parameters, while certain parameters can be overridden in parent files to target a specific vehicle or use case.

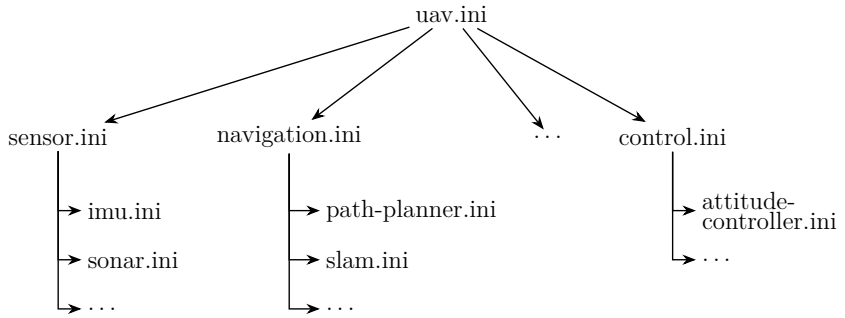


Figure 15.1: Configuration file structure for an example UAV

DUNE can be run in different profiles, which either alters the behavior of some tasks or choose which tasks are executed. A typical setup will usually include at least a simulation profile and hardware profile. Sensor driver tasks are started when the hardware profile is active, while the simulation profile will start simulator tasks instead. This allows the same configuration file to be used for both real missions and simulations, without altering the behavior of the system.

15.3 IMC

At the core of the LSTS Toolchain is the Intermodule Communication (IMC) protocol [49], which describes all data and messages being exchanged between different tasks, programs and systems. All message types are specified in an XML file named IMC.xml, which can be compiled into classes in C++ and Java, or used as object templates in Python. The IMC protocol is language and platform agnostic, and thus makes few assumptions about the data except for the structure defined in the XML file. The LSTS Toolchain already implements IMC for common use cases, but the user may also extend this with custom IMC messages if needed. An example IMC message can be seen in listing 15.1.

```
...

<message id="45" name="Desired Throttle" abbrev="DesiredThrottle">
<description>
  Desired throttle e.g. for Plane in FBWA-mode.
</description>
<field name="Value" abbrev="value" type="fp64_t" unit="%">
  <description>
    The value of the desired throttle.
  </description>
</field>
</message>

...
```

Listing 15.1: Example IMC message from the IMC.xml file

The IMC protocol facilitates a modular and encapsulated software design. An IMU sensor driver may read from a serial interface and publish *IMC::Acceleration* measurements to the IMC bus. Furthermore, an estimator uses the acceleration measurements along with other information, such as distance measurements, to create an estimated state of the vehicle, and broadcasts these as *IMC::EstimatedState*. These estimates can then be used by a controller to calculate output for actuators. One such example is a velocity controller broadcasting the desired thrust of an underwater vehicle using the *IMC::DesiredThrottle* messages. All subscription and broadcasting of IMC messages in DUNE are handled by the IMC bus. An illustration of the message passing described above can be seen in fig. 15.2

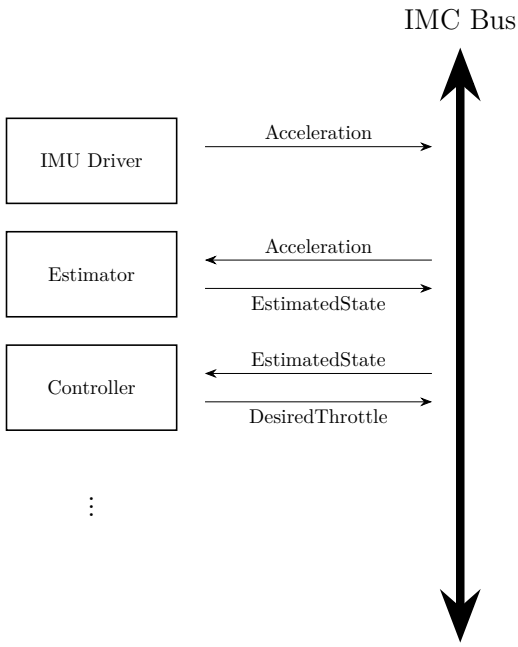


Figure 15.2: Message passing using the IMC bus

15.4 Neptus

The LSTS toolchain also includes a Ground Control System (GCS) called Neptus [59]. Neptus is intended as the main operator interface for vehicles developed using the LSTS toolchain. Neptus is written in Java, and hence runs on platforms that support the Java VM.

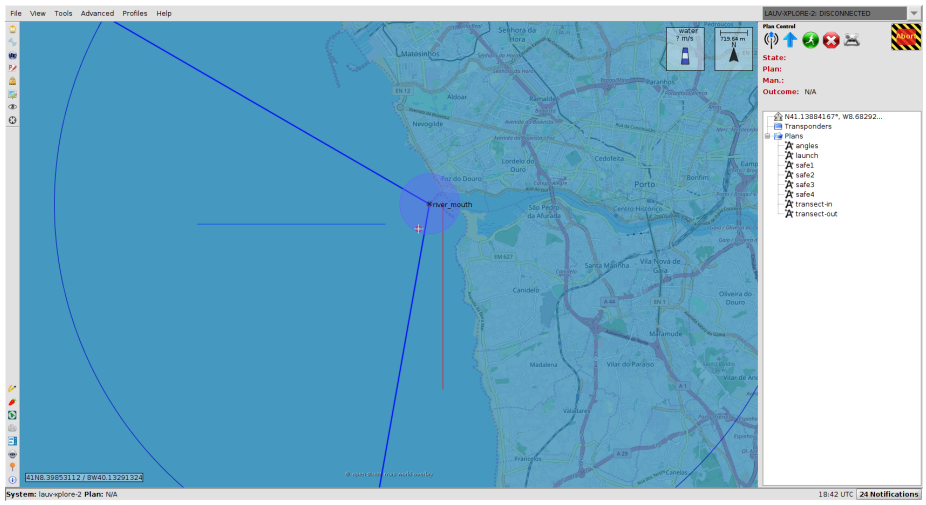


Figure 15.3: Neptus operation console for a UAV mission

All parameters and configurations running on the vehicle can be viewed in real-time, along with visualizations of flight data and plotting of any general time series. This gives the operator an overview of the situation, by showing all vehicles and entities involved, along with important mission data. Neptus is also able to change settings and display status for every task running on-board any of the vehicles. All communication with the vehicles is done over IMC. An example of a Neptus console is shown in fig. 15.3.

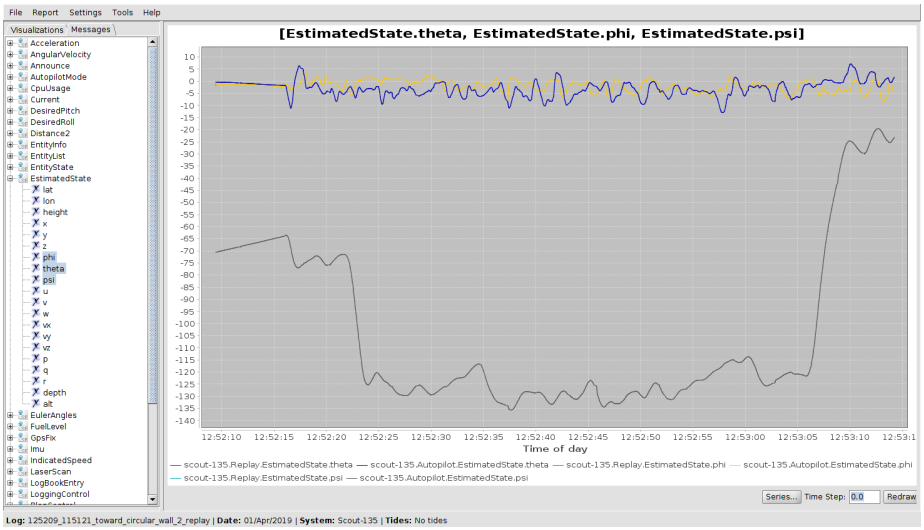


Figure 15.4: Neptus MRA. Roll, pitch and yaw from a test flight is plotted

The GCS also includes features for planing, executing and reviewing mission plans. Inspection and analysis of the mission data is done using a specialized application, called Neptus Mission Review and Analysis (MRA) [58]. After an operation is finished, Neptus can request a log of all messages sent and received by DUNE and display information back to the user. An example is illustrated in fig. 15.4, showing the estimated attitude of a drone during flight.

15.5 Linux Distribution

The operating system running on-board the drone is called the Ångström distribution, and is a light-weight Linux-distribution which aims to create a stable, user friendly Linux distribution for embedded devices. The distribution allows for configurations of the vehicle, such as hostname and network address, as well as specific programs or services to run at boot[2]. It should be noted that the Ångström distribution is not a part of the LSTS toolchain, contrary to the rest of the tools described in this chapter.

[2] http://www.angstrom-distribution.org/

Chapter 16

DUNE Implementation

This chapter presents the process of implementing the automatic thickness measurement on the DUNE software framework running on-board the UAV system. This first section will provide an overview of the entire system, to give the reader a broader view before details are presented in their respective sections.

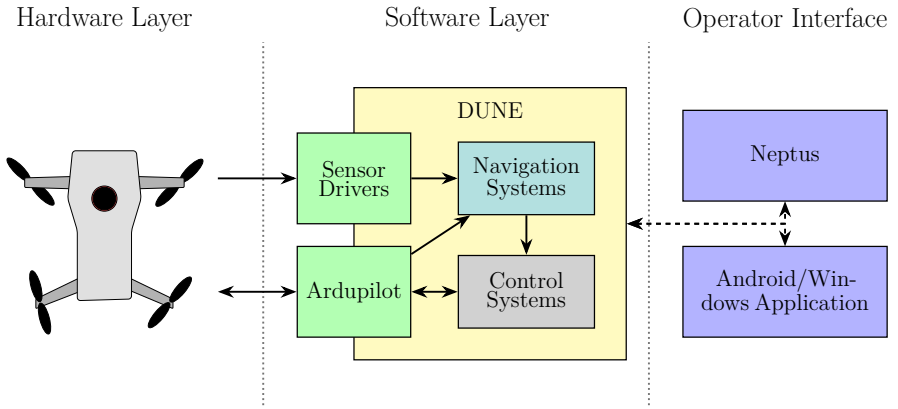


Figure 16.1: Visualization of the major components of the system

Figure 16.2 gives a more detailed view of the tasks and information flow between them. Tasks are visualized as rectangular boxes, and the arrows connecting the different modules in fig. 16.2 describe IMC messages dispatched and received by the different modules.

Since the creation of a complete software platform from scratch is beyond the scope of this thesis, some components developed by Scout DI have been reused. Section 16.1 will detail which modules this includes, but these tasks are also displayed with a dashed border in fig. 16.2.

The basis for almost all information in the software layer comes from sensors and mission payloads. In order to interpret the data received via the different hardware interfaces described in chapter 14, sensor drivers are needed. In fig. 16.2, these are showed in green, and section 16.2 will describe the drivers developed in this thesis.

As illustrated in fig. 16.1, the raw sensor measurements are transformed into navigation data by the navigation system in order to be utilized by the controllers. These are represented by the light blue boxes in fig. 16.2. The navigation system is an implementation of the navigation methods described in part IV, and will be detailed in section 16.3.

The two circles on the left side represents information sinks, rather then specific tasks. The blue circle is the operator interface, and represents the interfaces through which the user controls and operates the drone. In section 16.4, extensions made in this thesis to existing interfaces will be detailed. Finally, chapter 17 will provide a more detailed description of the integration of the control system, represented by the gray circle in fig. 16.2.

It should be noted that the information described in the following chapters is only intended as an overview, and is kept intentionally brief to not overwhelm the reader with the extensive amount of implementation details needed to develop the complete system.

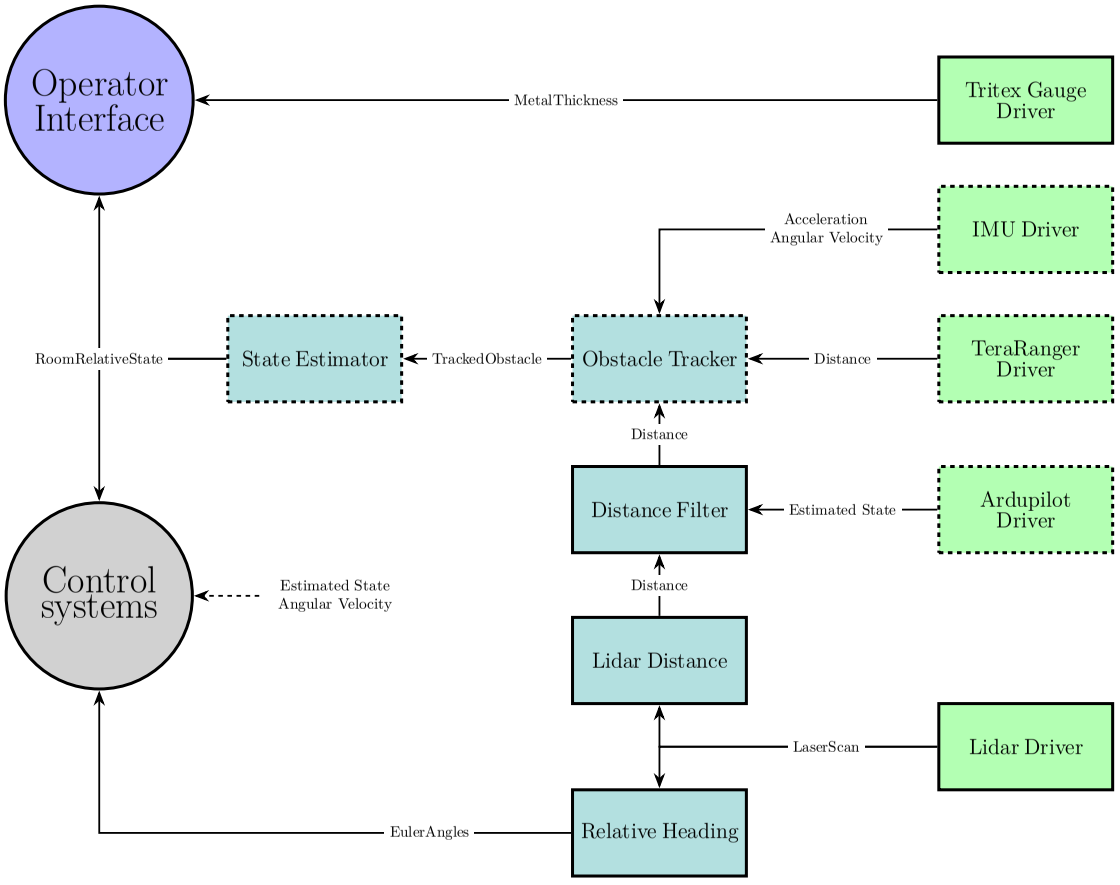


Figure 16.2: Information flow in sensor drivers and navigations system

16.1 Existing Framework and Modules

This section will describe the modules made available by Scout DI, and highlight what information these systems need to function correctly.

Scout DI has made many extensions to DUNE, such as improved logging and convenience functions for common operations that were utilized during this thesis. In addition to the core functionality, Scout DI also provided core modules for handling pilot inputs and custom commands to the Ardupilot software. Drivers for reading information from the downward facing TeraRanger and the Adis IMU was also made available.

Modules were also provided on the navigation side. As seen in fig. 16.2, this includes an obstacle tracker and a state estimator. The obstacle tracker receives distance measurements to obstacles in the horizontal plane and fuses it with IMU measurements to create a tracking filter. These tracked obstacles are then combined in a state estimator to provide relative distances to obstacles in 5 sectors around the drone (front, left, right, back and down), as visualized in fig. 16.3. The *IMC::RoomRelativeState* from the state estimator form the basic control variables used in all control systems. Tuning and adjustments of the obstacle tracking filters has been a part of this thesis.

Control systems with functionality for take-off, landing and pilot assisted flight were also made available. These will be described in more detail in chapter 17.

16.2 Sensor Integration

This section will present the sensor driver developed as a part of this thesis. This includes software interfaces for the RPLIDAR (see section 14.6) and the Tritex thickness measurement probe (see section 14.5).

16.2.1 2D LIDAR

The 2D scanning LIDAR used in this thesis comes with an SDK to handle the low level-communication over the serial interface. This SDK was included in the DUNE framework, and a task was created to handle all communication through the SDK, to encapsulate all low-level communications.

A custom IMC message named *IMC::LaserScan* was created to transmit the LIDAR measurement, modeled after a similar message in ROS[1]. The message includes information about the start and angle increments between measurements, in addition to the raw range measurements.

It should be noted that the LIDAR does not provide any timing information about the laser scans, and hence all timestamps are generated in the driver. This is problematic if high accuracy scan information is needed, for example for transforming points using the attitude of the drone during acrobatic maneuvers.

A simulator was also created to mimic the behavior of the LIDAR in a rectangular room, which gives the possibility to test the LIDAR behavior in combination with a flight dynamic simulator from Ardupilot.

16.2.2 Thickness Measurement Probe

The thickness measurement probe exposes a serial interface via an RS232 cable. The protocol used for this serial interface is not public; however, documentation was provided by Tritex NDT such that it could be utilized. The probe can transmit either raw timing measurements or actual thickness measurements if the probe has been calibrated with the correct speed of sound for the material of interest. The probe is pre-calibrated for steel by the manufacturer, thus the transmitted measurements are based on this.

The probe alternates between sending two different package types every 250 ms. The first package is a status packet, containing information about the probe type, firmware version and other practical information. The second type is the measurement package, containing either an invalid reading or the thickness of the material measured. The downside to this approach is that measurements are only sent 0.5 s, which is suboptimal in case of a short contact. The exact timing of the measurement appears to be linked with the timing of the message sending, which means that the probe needs to have good contact exactly before the message is sent. After the experimental results in this thesis was finished, Tritex shipped a new probe with a firmware version that allows only measurement packages to be sent. Hence, measurements are sent every 250 ms, which increases the chance of measurement during shorter contacts. The driver was updated to support this option as well.

A custom IMC message named *IMC::MetalThickness* was created to transmit the result of the thickness measurement.

[1] http://docs.ros.org/melodic/api/sensor_msgs/html/msg/LaserScan.html

16.3 Navigation systems

The following sections will present the navigation tasks in fig. 16.2. The existing framework of obstacle trackers described in section 16.1 works by tracking obstacles in sectors around the drone, as shown in fig. 16.3. These tracked obstacles are then combined to produce *IMC::RoomRelativeState*, which all control systems use as input. The trackers need distance measurements to detected obstacles in order to fuse this information with the IMU data. This is provided by the task described in section 16.3.2, but also filtered though the task described in section 16.3.3. An alternative to the above solutions is to use the distance measurements directly; however, the LIDAR only provides measurements at 10 Hz. Thus fusing it with IMU data at 100 Hz allows for both filtering of the distance signals and providing distance estimates at a higher rate to the control system.

In addition, a relative heading estimate for the forward sector was required to align the drone to the inspection surface. This was provided by the task described in section 16.3.1.

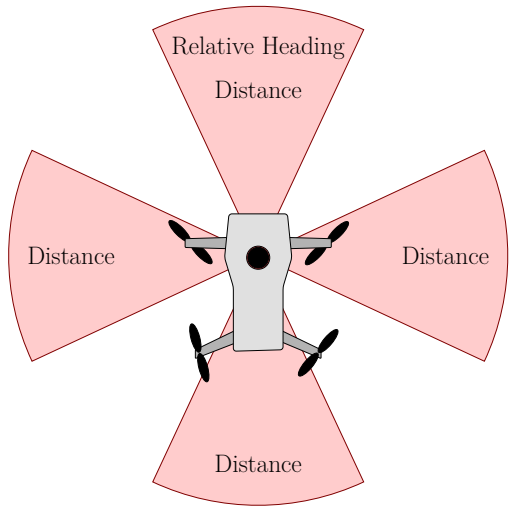


Figure 16.3: Visualization of the tracked sectors. Downward sector is not shown

16.3.1 Relative Heading

Receives the *IMC::LaserScan* information from the LIDAR driver, and dispatches relative headings to the surface in front of the drone. This is the implementation of the methods for relative heading estimation developed in section 13.3 and section 13.2. Configuration files decides which of the methods that should be run; however, the RANSAC method is default.

16.3.2 LIDAR Distance

Receives *IMC::LaserScan* information from the LIDAR driver, and dispatches distances to the closest surface in all four horizontal sectors.

Two different solutions where implemented: averaging a few measurements in the center of the sector and detecting surfaces using the RANSAC and regression methods from chapter 12. The current implementation is chosen in the configuration files. While averaging range measurements in the center of a sector makes no assumptions about the surroundings, it fails to provide rejection to faulty measurements and sensor noise. Hence, running RANSAC gave the most robust performance, and in the results shown in part VII this is the active implementation.

16.3.3 Distance Filter

As mentioned, RADARs provide distance measurements in the horizontal plane, even when the drone is tilted. Hence, the obstacle trackers expect to get the distance in the horizontal plane to nearby obstacles. However, the LIDAR data violates this assumption since pitch and roll will affect the distances. To counter this effect, two options were considered: assume the detected surface is flat in the vertical direction and compensate using geometrical considerations, or discard the measurements and let the obstacle tracker provide an estimate only based on the last known distance and new IMU data. Based on experimental tests, a choice was made to discard measurements during these periods, since this proved to give accurate estimates, while making fewer assumptions about the vertical geometry of the environment. Hence, the the Distance Filter task filters out measurements based on the current attitude of the drone.

16.4 User Interface

An important part of creating a robust and complete system for industrial use is providing easy and intuitive user interfaces for different use cases. To fulfill this requirement, interfaces for two different use cases were implemented as described in the following.

16.4.1 Neptus Plugins

Two plugins were developed for Neptus, in order to best integrate with the existing command and control options in th LSTS toolchain. One plugin displays the current (if any) thickness measurements, along with the last measurement and time since the last measurement. This mimics the behavior of the user interface software that was bundled together with the probe. In addition, a plugin was created to give easy access to *IMC::OperatorCommand* with *Arm*, *Takeoff*, *Land* and *ThicknessMeasurements* as value. The plugins can be seen below the attitude HUD in fig. 16.4.

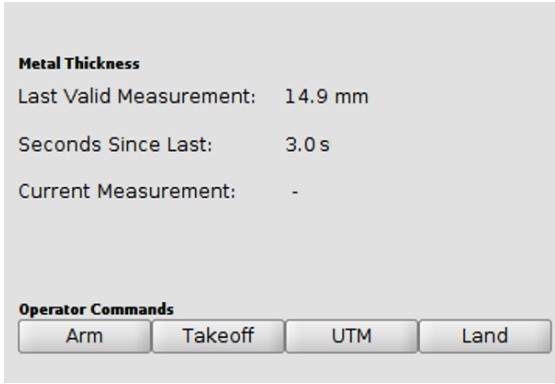


Figure 16.4: Screenshot of the two plugins created for Neptus

16.4.2 Mobile Application

Scout DI has developed an android application that is intended as the main way to control and communicate with the drone. The application, shown in fig. 16.5, features buttons for arming, take-off and landing, in addition to joysticks for controlling the drone. In the upper right corner a visualization of the attitude and proximity to nearby obstacles is shown.

The application is developed in a framework called FUSE. Fuse introduces a declarative XML-based language for creating components for iOS and Android called UX markup. UX Markup compiles down to C++, which gives optimal performance on native, mobile devices. This is achieved through Uno, a dialect of C# that compiles to C++. Hence, FUSE is a multi-platform framework and supports building for Android, Windows and iOS.

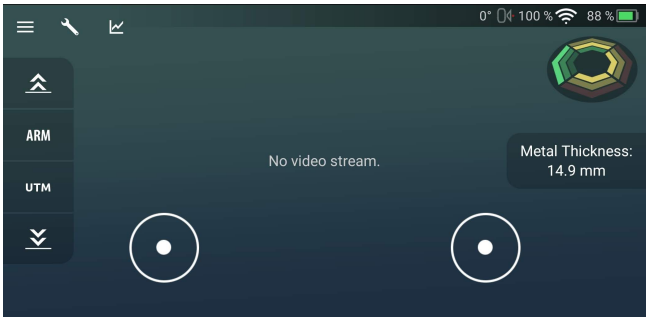


Figure 16.5: Screenshot of the application. UTM button on the left and thickness measurement display on the right

As a part of this thesis, a button for starting the thickness measurement was added to the application, as seen by the third button on the left side of the application in fig. 16.5 (UTM Button). The application will prompt the user for confirmation, before sending the command to the drone and start the operation. In addition, a display for valid thickness measurements was integrated. Valid measurements are transmitted back to the application, and displayed live to the operator (seen on the right side in fig. 16.5).

By adding these functionalities to the application the thickness measurement operation is provided as a one-button solution to the inspector. This empowers even basic drone operators with the capabilities to perform advanced inspection operations, and lowers the requirements for pilot training of inspectors.

Chapter 17

Control System

This chapter will explain the implementation of the impedance controllers developed in part III within the DUNE framework. This is done by integrating the thickness measurement controllers and maneuvers into an inspection plan provided by Scout DI.

17.1 Introduction

A typical DUNE operation is implemented as a plan, and consists of several maneuvers. Maneuvers can be anything from take-off and circular loiters to more complex area covering patterns. Each of these maneuvers may activate several control loops in order to perform the desired maneuver, resulting in a layered control structure. A visual representation of this hierarchy can be seen in fig. 17.1.

A plan is implemented as a state machine where each state is a maneuver. Each transition between maneuvers have specific triggers and required conditions, for example receiving an *IMC::OperatorCommand* or reaching a certain target set-point.

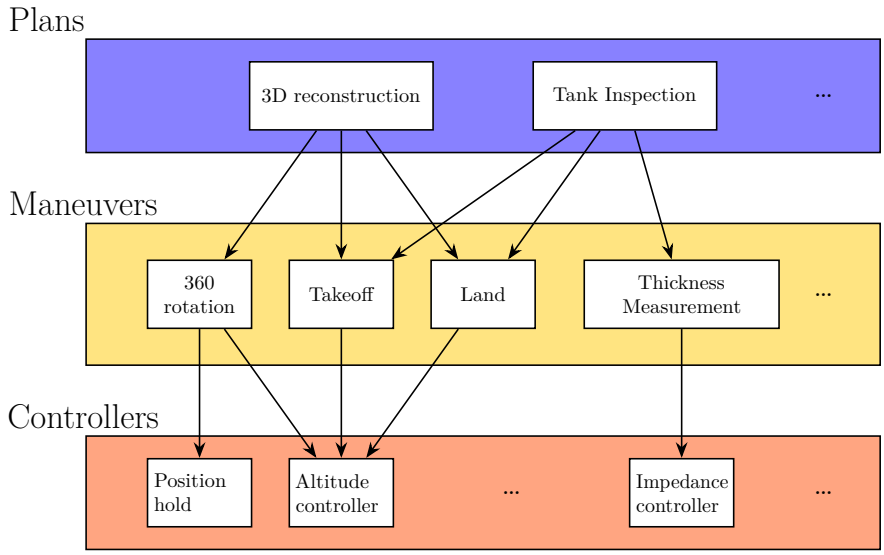


Figure 17.1: Example DUNE control hierarchy

17.2 Plan Integration

To integrate thickness measurements into the existing inspection scenario, a thickness measurement maneuver and an impedance based control loop was added to the existing inspection plan. This incorporated the controllers developed in part III into the DUNE framework. A visualization of the entire inspection plan can be seen in fig. 17.2. The trigger for initializing the operation was set to an incoming *IMC::OperatorCommand* with a value of ThicknessMeasurementRequest, while the trigger for transitioning back to pilot assisted mode was an *IMC::ManeuverComplete* message from the controller. This enables the operator to easily take-off, fly to the desired position in the assisted mode, and with one button activate the thickness measurement at a surface of interest.

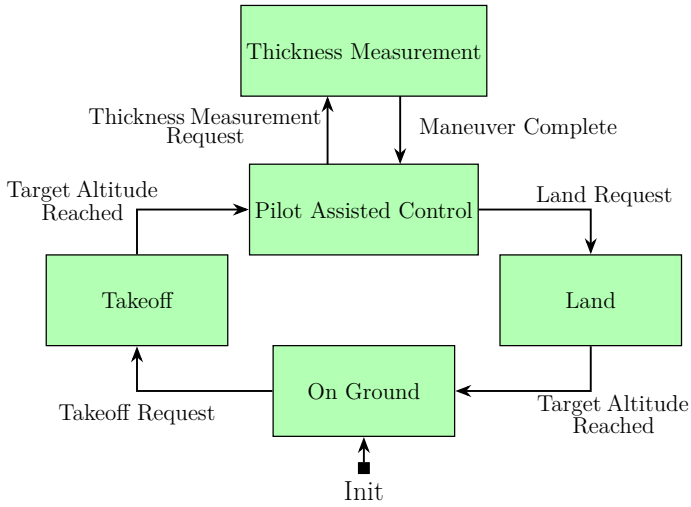


Figure 17.2: DUNE Plan

17.2.1 Controller Implementation

As mentioned, the impedance controllers developed in part III was implemented as control loops in DUNE. The information received and dispatched by the impedance controller is visualized in fig. 17.3, along with the source and destination of the information. The output from the controller is dispatched to the low-level attitude controller using the *IMC::DesiredCopterControl* message, containing the desired attitude, climb-rate and yaw-rate.

As discussed in chapter 9, the impedance based controllers works by setting position based set-points for the controller. Hence, the thickness measurement maneuver works by dispatching a sequence of set-points to the controller. The first set-point is the location when the maneuver was activated, but with a desired relative heading of zero, thus causing the drone to align itself to the inspection surface. After the alignment phase, the drone choses a new set-point close to the inspection surface. When the drone arrives at the target set-point, contact is initiated by setting a new set-point beyond the surface. This set-point is held for a fixed duration, which can be changed in the configuration files. After the contact phase is done, the drone returns to the initial set-point and dispatches *IMC::ManeuverComplete*.

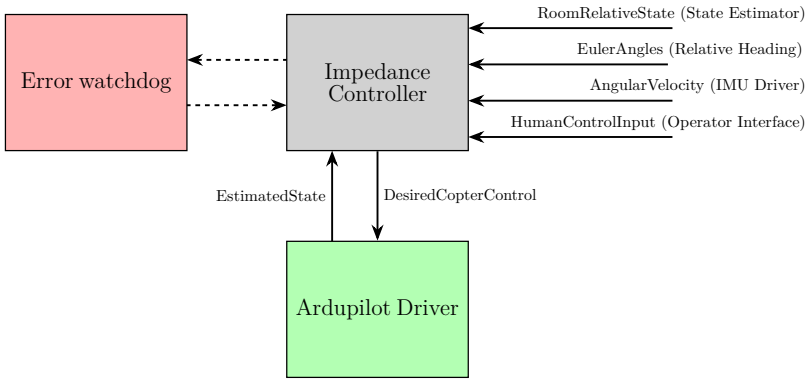


Figure 17.3: High-level visualization of information flow around the controller

Coordinate System

As mentioned in part IV and the sections above, the navigation systems provide local information about the surrounding environment. When the thickness measurement operation is started, this results in the controllers operating relative to a coordinate system as shown in fig. 17.4. The coordinate system will be local to the environment where the operation is started, and hence changing for every thickness measurement operation. However, since the coordinate system is aligned with the inspection surface, distance measurements are only valid under the assumption that $\psi_{\text{rel}} \approx 0°$, as deviations in yaw will affect the measured distance. Hence, if large deviations in relative heading is experienced, the thickness measurement will be aborted as described in the following section.

The information received by the state estimator also contains a sequence number for the current tracked obstacle. While it is possible to use this information to track position in highly changing environments, it is difficult to guarantee data accuracy in the switching phase, and thus a decision was made to abort the thickness measurement in this scenario.

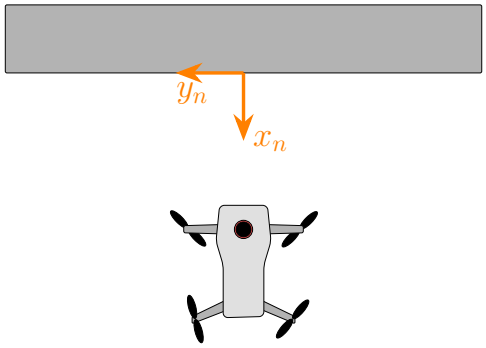


Figure 17.4: Local coordinate system. Z-axis pointing down

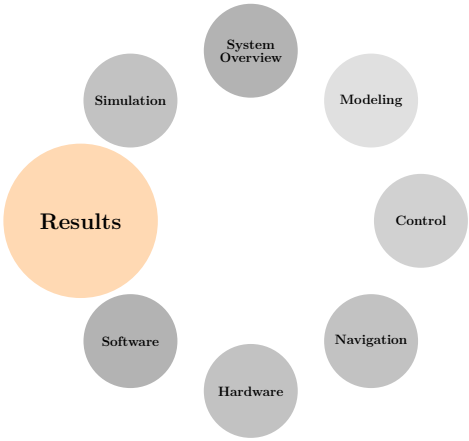
Error Handling

One of the goals for this thesis was to create a robust and user friendly system for inspection scenarios in an industrial environment. Hence, an effort was made to ensure safe operation of the system at all times. The combination of the high kinetic energy of a flying drone with contact-based interactions, imposes severe risk to both the drone and its surrounding environment.

To comply with the desire for safe, a number of features were implemented to track the progress during the interaction. Timers were included to detect loss of navigation data, in addition to monitors for breach of operation limits, such as maximum pitch or heading during different stages of the docking phase. The implementation was done such that any violations of the security constraints results in the drone returning to its original position and give handling control of the vehicle to the operator. Any input from the operator will immediately cause the controller to return to pilot assist mode, and thus yielding control back to the operator.

Part VII

Results



Chapter 18

Results

This chapter will present the results achieved in this thesis on both the development platform detailed in part V and the production platform described in section 14.7. First, the different test locations will be described in section 18.1, before the results from the following scenarios will be described:

- **Flight Test I** - *Test Room*
  This flight test shows the result of autonomous thickness measurements in a test room.
- **Flight Test II** - *Falck Nutec Field Test*
  This section shows the result of experiments conducted at the Falck Nutec test facilities.
- **Flight Test III** - *Automatic Abort*
  This flight test showcases the automatic abort system. Conducted in the same test room as Flight Test I.
- **Flight Test IV** - *Simulation Comparison*
  A side-by-side comparison of Flight Test I and simulations carried out in Appendix Part I.

For each scenario presented in this chapter, a video is also attached. These can be reached by scanning the QR-codes or by following the link below the QR-code. The videos are also attached in the digital appendix following this thesis. The reader is highly recommended to watch these videos, as they are essential to get the full understanding of the drone performance and interaction of the complete system as seen from the perspective of the operator.

In the following, the inspection surface is defined to be parallel with the y-direction, and hence the terms ”parallel to the inspection surface” and ”perpen-

dicular to the inspection surface" is used interchangeably with "y-direction" and "x-direction" respectively. This is to simplify the wording in the discussions, and is of no other significance to the results. In addition, the phrase "angle-of-attack" is used to describe the angle between the probe and the vector perpendicular to the inspection surface at impact.

18.1 Test Facilities and Setup

This section will briefly describe the different facilities used for flight testing.

18.1.1 Test Room

A test room was setup in order to facilitate rapid testing iterations of the drone. To represent the tank wall being measured, a steel plate was mounted in an aluminum wall segment, as seen in fig. 18.2. However, the aluminum plates was too thin to give a valid measurement, so the drone has to touch the steel plate for the probe to register a reading. The room is square, and the walls range from metal to concrete with different textures and colors, and gives an indication of the navigation sensor performance on different materials. The control station is protected by a net to ensure the safety of the operator. In addition, the floor is padded with mattresses to minimize the consequences of a potential crash.

Figure 18.1: Overview of the test setup in the test room

The test plate was measured to be between 11.9 mm and 12.0 mm using manual measurements. Since the development drone did not have wireless data transmission, an Ethernet cable was used to transmit data to mobile application of the operator and debug data to the ground station in the flight test described

in section 18.2. For the Scout 135 platform only a tethered power supply was used to extend the flight time, as this platform featured wireless transmission capabilities.

18.1.2 Falck Nutec Tank

Test have also been carried out at the Falck Nutec facilities[1] in Trondheim. The tank is normally used for safety courses for industrial purposes; but it is also ideal for testing realistic inspection scenarios. Only the newer hardware platform was tested in this environment, where a tethered power supply was used to extend the flight time possible.

While the steel plate in the test room is smooth with no visible deteriorations of the material, the tank surface is rougher, with visible corrosion. Hence, it provides a more realistic test environment for the measurement probe. Manual measurements varied between 6.0 mm and 6.5 mm, depending on the exact area of the tank. However, it was very difficult to get valid results from the probe, even with manual measurements. This will be discussed in more detail in the summary of this chapter (section 18.6).

As described in section 14.6, the LIDAR was not performing as expected in this environment. Hence, tracking was not available in the y-direction, and so the plan was to forward roll commands from the pilot to counter any drift along this axis. However, this turned out to be unnecessary, as just keeping the roll angle stabilized yielded satisfactory results.

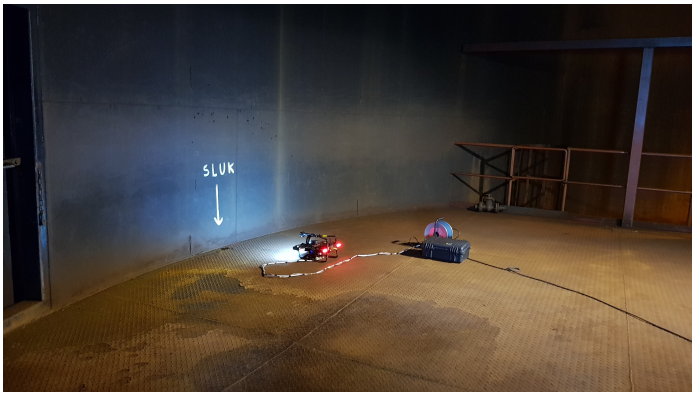


Figure 18.2: Overview of the test setup at Falck Nutec facilities

[1] https://relyonnutec.com/no_no/kurslokasjoner/?locid=1507

18.2 Flight Test I - Test Room

The following section will describe the results from flying the development platform in the test room. The video can be seen by following the QR-code or link in fig. 18.4. In the upper left corner the live navigation data is displayed, along with the surfaces found by the RANSAC algorithm for all four sectors around the drone. In the lower right corner the operator mobile application is shown. For safety reasons, stick input is given from another controller, and hence is not visible in the mobile application.

Figure 18.3: Snapshot from the video showing the contact and valid measurement

Figure 18.4: QR-code for the flight test in the test room
URL: https://youtu.be/-eNShie2HE4

During takeoff and landing the drone is not stabilized in position, which causes it to drift slightly due to disturbances and imbalances. After takeoff the drone enters pilot assistance mode, and the operator positions it in front of the area of interest. Thereafter, the thickness measurement button is pressed and the drone performs an autonomous thickness measurement.

First, the drone aligns itself to the inspection surface. This can be seen in the live navigation data, but also in the bottom graph of fig. 18.5. The relative heading starts at 5°, and is quickly corrected towards zero. After the alignment phase, the drone sets a new set-point close to the inspection surface and starts approaching it. During the approach phase, the relative heading is varying slightly; however, it stays within ±2°. When the drone arrives at the target set-point, contact is initiated by setting a new set-point beyond the surface. During the contact the pitch stabilize around 4°, with slight oscillations in the transient phase. Both the roll and relative heading remains stable during the time of contact. The contact phase last for approximately 2 s, and a valid thickness measurement of 11.9 mm is displayed on the right side of the operator application. A visualization of the steady state contact can be seen in fig. 18.6a. Thereafter, the drone returns the initial position, and control is given back to the operator. The estimated position in the xz-plane during the entire flight can be seen in fig. 18.6b. The plot is created by using the estimated distance down and forward.

Right before the drone comes in contact with the surface, a drop is seen in the relative heading (bottom graph fig. 18.5). This is most likely due to aerodynamic disturbances caused by irregular airflows close to the inspection surface. These wall and ground effects are difficult to predict, but controllers quickly compensate for it before the drone comes in contact with the inspection surface.

To summarize, the above flight test shows a successful thickness measurement operation. The navigation methods provide data with enough accuracy for the controllers to successfully perform a stable interaction and get a valid measurement from the probe. The repeatability of the thickness measurements are also good. During all of the test conducted during development and tuning of the controller, no unstable contacts occurred. When the probe touched the target steel plate, a valid measurement was given about 90% of the time, mostly depending on the amount of gel applied. This could potentially be improved by extending the contact period beyond the 2 s used in this flight test.

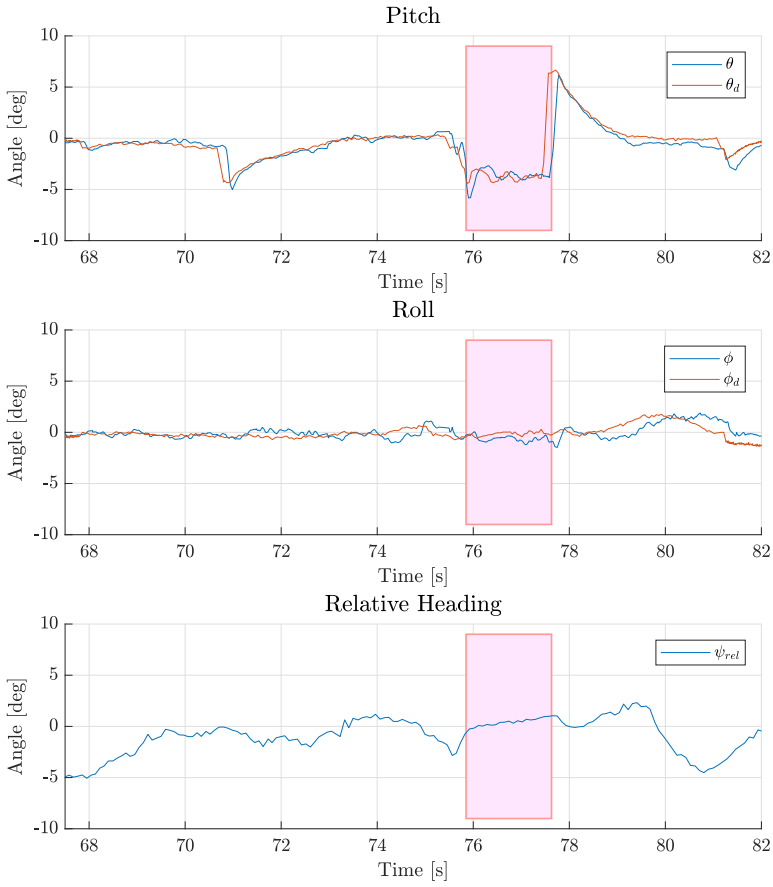
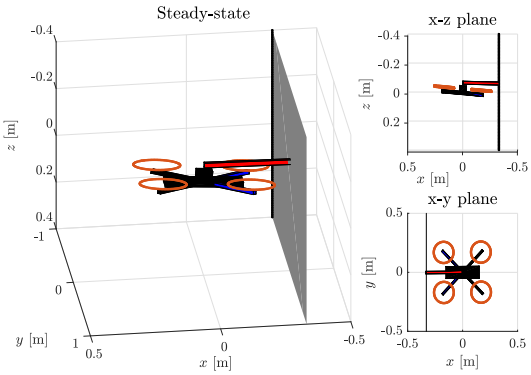
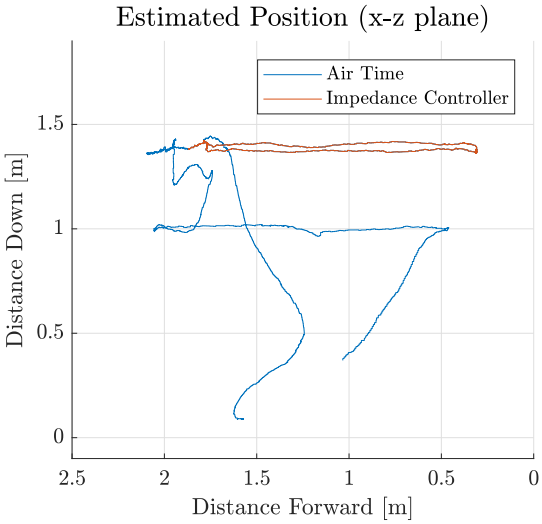


Figure 18.5: Attitude during the flight test in test room. (Contact period marked in red)



(a) Visualization of contact steady-state from flight test



(b) Estimated position during the flight test

Figure 18.6: Flight test in test room

18.3 Flight Test II - Falck Nutec Field Test

The following section will describe the results from flying the Scout 135 in at the Flack Nutec test facilities. The video can be seen by following the QR-code or link in fig. 18.8. On the left side, a sideways view of the interaction can be seen, while the right side shows the operator perspective. The live navigation data, along with the surface found by the RANSAC algorithm for the forward section, is display in the middle. During this flight test, stick input is given from a gaming controller attached to the operator tablet.

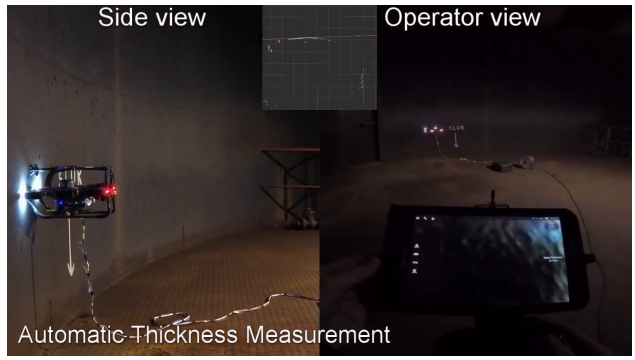


Figure 18.7: Snapshot from the video showing the contact and valid measurement

Figure 18.8: QR-code for the flight test at the Falck Nutec facilities
https://youtu.be/Uw_suetKRPc

After takeoff the drone enters pilot assistance mode, and the operator positions it in front of the area of interest. In the test flights performed at the Falck

Nutec facilities, a heading controller was activated in the pilot assist mode, using the same navigation data from the methods developed in this thesis, as well as the same control technique. This allowed the drone to remain locked-in on the inspection surface, both with respect to the distance and orientation. Hence, the operator can perform a sweeping maneuver along the wall by only controlling the roll angle of the drone. This is a very useful maneuver during visual inspections, enabled by the developed navigation methods. However, this means that the drone was already aligned when it initiated the thickness measurement, as seen in the third graph in fig. 18.9.

The drone follows the same procedure as in the previous section by first setting a set-point in front of the wall, before initiating contact by putting a set-point behind the surface. In this flight test, the contact time was adjusted to be half a second longer, in order to increase the probability of a valid reading on the rough surface. The contact phase lasts for approximately 2.5 s, and a valid thickness measurement of 6.1 mm is displayed on the right side of the operator application. A visualization of the steady state contact can be seen in fig. 18.10a.

About 0.5 s after the contact is complete, the operation is canceled by the operator touching the stick. This can be seen by the rising edge in the bottom graph in fig. 18.9, showing the pilot input along the x-axis. Hence, the drone did not return to the initial position as seen in fig. 18.10b, but control was given back to the operator as intended.

Tuning the impedance gains was easier on the new hardware platform compared to the development platform. The robust hardware design means that a wider range of parameters resulted in a stable contact. The support from the frame of the drone allows for better disturbance rejection, and especially for disturbances in yaw. When the drone is in contact, the behavior in the xy-plane is similar to an inverted pendulum. The more force the drone applies to the inspection surface, the more effect a small disturbance will have in deviations in yaw. Hence, having a hardware platform that helps stabilize allows for more force in the interaction.

An interesting observation was also made with respect to the behavior of the drone platform in the tank environment compared to preliminary tests conducted in the test room. There were noticeably less disturbances in the tank, most likely due to the more open environment. While the controllers were able to successfully complete the operation in both environments, less compensation had to done in the tank environment.

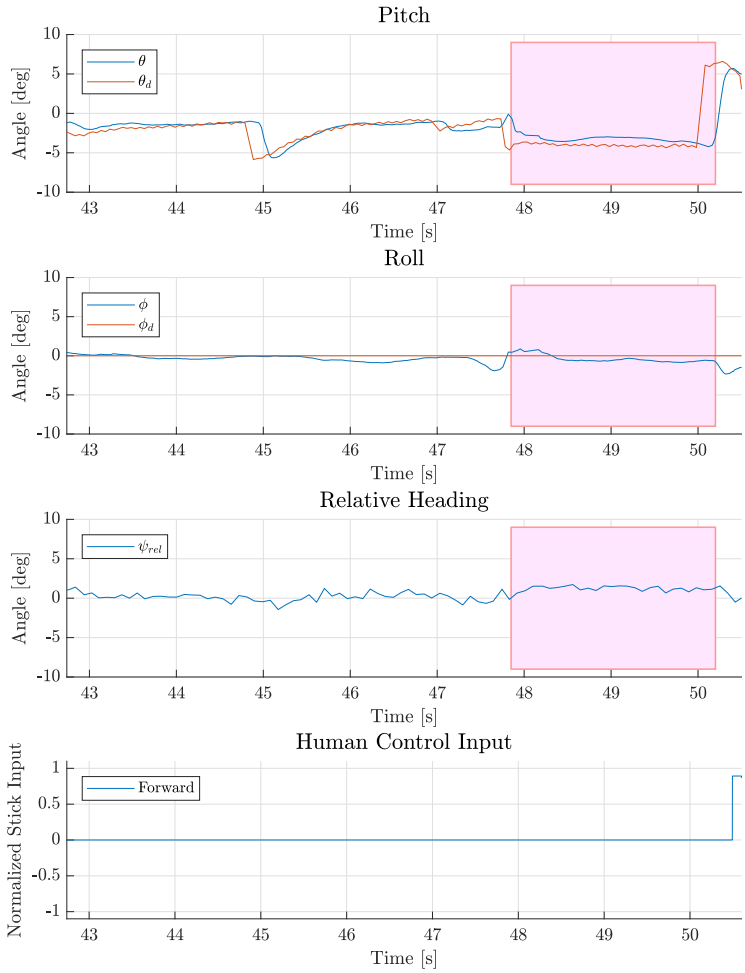
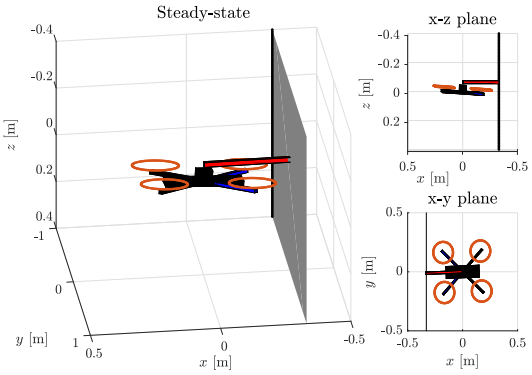
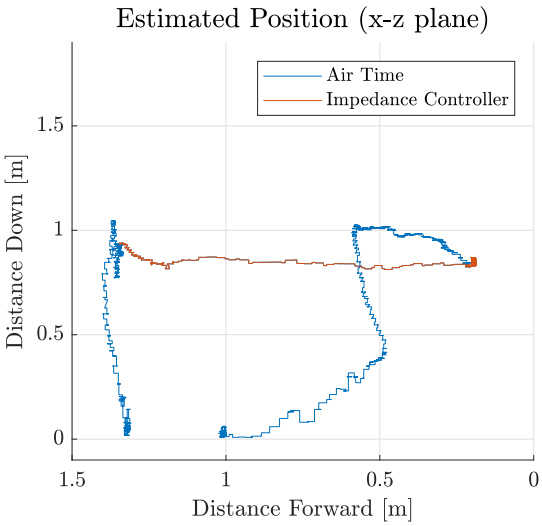


Figure 18.9: Attitude during the flight test at the Falck Nutec facilities for the duration of the thickness measurement. (Contact period marked in red)



(a) Contact steady-state from flight test



(b) Estimated position during the flight test

Figure 18.10: Flight test at the Falck Nutec facilities

18.4 Flight Test III - Automatic Abort

This section will showcase the automatic abort system that was integrated along with the controllers. The video can be seen by following the QR-code or link in fig. 18.12. On the upper right, the live navigation data along the surfaces found by the RANSAC algorithm is displayed, while the operator application is shown on the lower right side. A thread was attached to the front left side of the drone. Hence, pulling the thread would cause the drone to rotate, simulating an unknown disturbance.

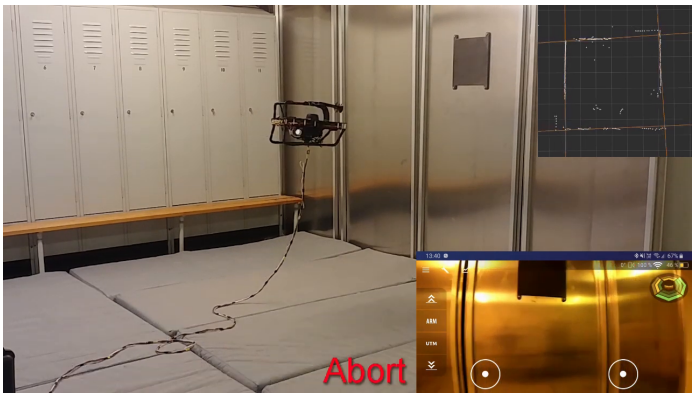


Figure 18.11: Snapshot from the video showing the moment of abort

Figure 18.12: QR-code for the flight test showcasing the automatic abort system https://youtu.be/T0ZzDSyEA5I

As in the previous sections, the drone takes off and the thickness measurement is initiated. However, as the drone approaches the inspection surface, the thread is pulled and the drone rotates. As seen in the bottom graph of fig. 18.13, this causes the relative heading to deviate beyond the set limits. A visualization of the attitude at the moment of abort can be seen in fig. 18.14a. The operation is aborted and the drone returns to the original position as seen in fig. 18.14b.

The first and second graph in fig. 18.13 shows that also the pitch and roll is affected by the large disturbance caused by pulling the thread. However, the system quickly stabilizes and enables a safe return. In general, it can be seen both in this test and the previous section (section 18.4), that the attitude response is better for the Scout 135 platform compared to the development platform. This is especially true for the pitch angle. This is due to the Scout 135 having a more even distribution of weight, meaning it is easier for the low-level attitude controller to keep the drone balanced.

Violation of other operational limits or timeout on important navigation data will also trigger the same response and return the drone to its original position. If navigation data about the position of the drone is lost, the system will return to pilot assist in order to yield control to the operator.

It as also worth noticing that during this flight test the power cable was not completely unwound, and thus the drone has to counteract the disturbance force caused by dragging the heavy cable along the floor. The impedance controller reacts to this in the same manner as it would with environment contact disturbances, and enforces a constant force while it pulls the cable. This shows the impedance controllers ability to maintain stable interaction without causing any build-up of interaction forces, even in the presence of an unintentional interaction force.

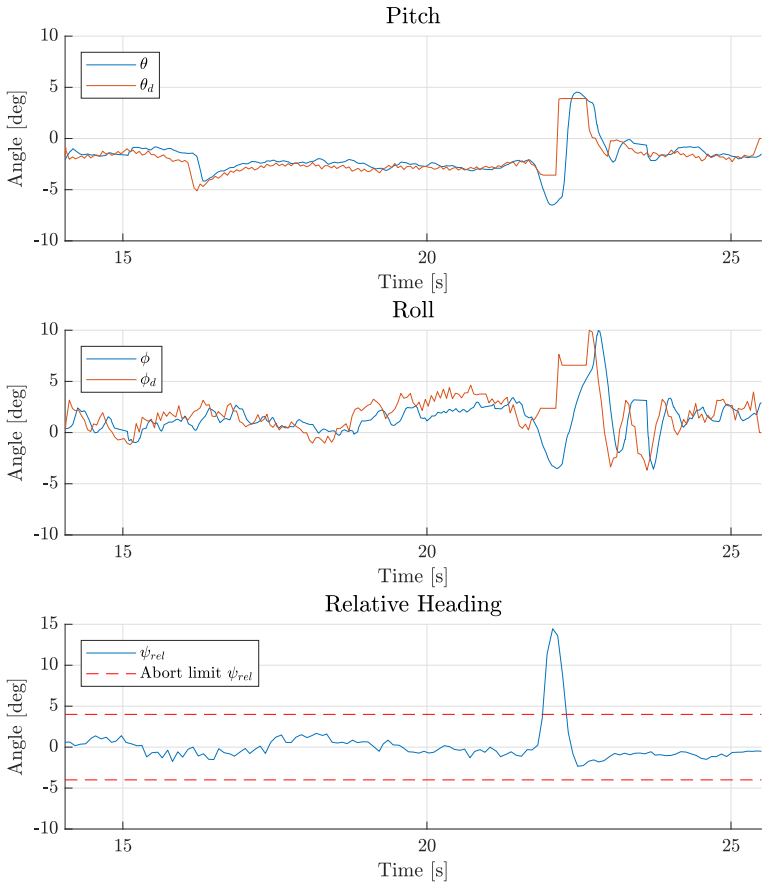
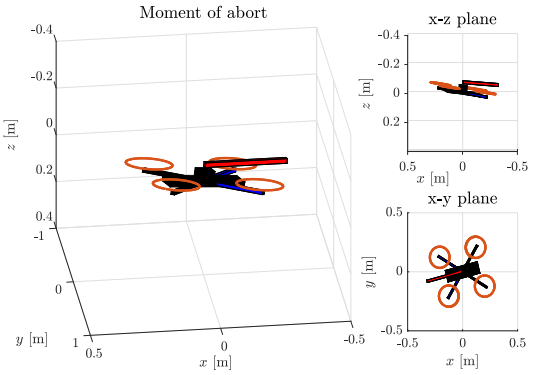
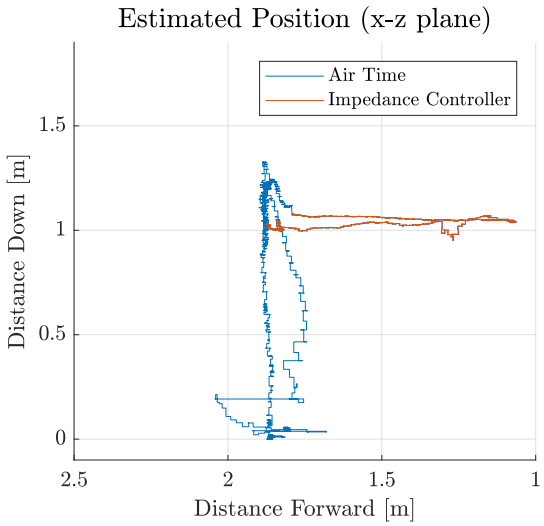


Figure 18.13: Attitude during the flight test showcasing the automatic abort system



(a) Contact steady-state from flight test



(b) Estimated position during the flight test

Figure 18.14: Flight test showcasing the automatic abort system

18.5 Flight Test IV - Simulation Comparison

The following section contains a video showing a side by side view of the flight test from section 18.2 and the same controller implemented in simulations. This gives a perspective of the key differences between simulations and real life experiments, but also evaluates the performance of the physics engine simulation from Appendix Part I. The video can be seen by following the QR-code or link in fig. 18.16.

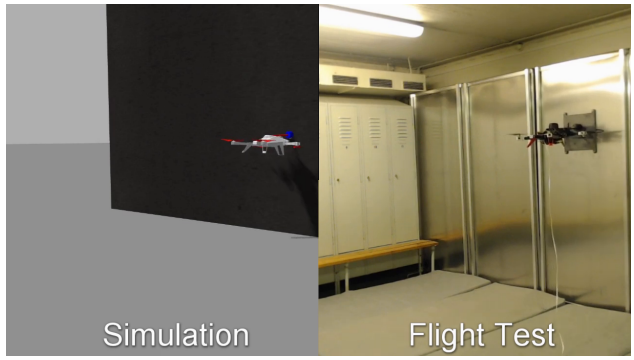


Figure 18.15: Snapshot from the video showing the contact

Figure 18.16: QR-code for the simulation comparison
https://youtu.be/w8xs4e4oM3I

From the video it is very clear that the simulation model does not experience the same level of disturbances as in the actual flight test. None of the models in this thesis considers the effects of changed aerodynamics when the propellers are in the proximities of surfaces, as the spinning propellers will create unpredictable vortexes. Work has been done to model these effects, such as [50], where wind tunnel experiments were carried out on axial-flow rotors. However, these effects are highly turbulent and very difficult to model accurately as they heavily depend on the geometry of the surrounding surfaces and complex fluid simulations. A possibility is to simulate these disturbances as wind; however, this is not entirely realistic. This approach will not capture the spatial correlation to the direction and magnitude of the disturbance, but at least it gives the opportunity to test the control system with aerodynamic disturbances.

Since the disturbances mentioned above are not present in the simulation, the controllers and set-points may be tuned more aggressively. However, when the system was implemented on the hardware platform, it was apparent that rapid movements close to any surfaces causes irregular airflows. These disturbances are hard to counteract for the control systems, especially during the contact phase. The best way to counter this problem, is to have slower movements, and thus fewer radical changes in the airflows. Hence, the operation is slightly slower in the flight test, as seen in the video.

The compliance device in the simulation provides more damping compared to the compliance device mounted on the hardware platform. While the behavior of the compliance device can be tuned stiffer in simulations, some of the differences is a result of the ideal spring simulation, which is not possible to replicate in hardware. All of these differences allows the compliance device to handle more interaction forces in the transient phase of the interaction, as seen by the steeper entrance angle in the simulation.

Overall, the simulations provide a very realistic test environment for the interaction scenario. The experimental validation of controllers for interaction is a major hurdle in the development process. A crash can severely damage the hardware of the UAV, increasing the cost and potentially delaying the release. Hence, the extension to existing simulation frameworks carried out in this thesis enables faster testing of contact-based inspection scenarios. However, the aforementioned differences should be kept in mind, as to not overfit the controller to the simulated scenario.

18.6 Summary

This section summarizes the main points from the experimental validation of the controller, navigation method and overall system.

Successful experiments have been carried out on multiple platforms. The operations allows the inspector to fly to an area of interest, press one button, and the drone will automatically perform a thickness measurement and present the result back to the operator. While the results only show individual experiments, the operations have been repeated many times during the development and testing of the system. Not once did the contact operation lead to a crash. This shows the robustness of the developed controllers.

Regarding the repeatability of valid measurements, it was apparent that this depended heavily on the surface. On the test plate it was successful about 90% of the time. However, in the tank environment the success rate was significantly lower. It turned out that it was very difficult to get valid results from the probe, even with manual measurements. Hence, the poor performance can not be contributed to the controllers, but instead depended on the probe itself. The probe used in these experiments is an early developer version of the product, and newer versions will hopefully improve the consistency of the measurements.

One of the major weaknesses of the impedance based controllers is the need for set-point manipulation. This was discussed in the simulation results (chapter C); however, during the testing of the controllers this revealed to be easier compared to the simulated environment. The use of two set-points, one close to the surface and one to initiate the contact, proved to be an adequate method for selecting these points. The force during the interaction could easily be adjusted by moving the second set-point further or closer from the back of the inspection surface.

When it comes to disturbance rejection by the controllers, the performance depends on which axis is affected. Parallel to the surface, the controller is more sensitivity to noise during the interaction. This is natural since changing the roll angle to fix the positional error is not possible as the motion is constrained. An easy way to mitigate this problem is ignoring the error parallel to the surface during the interaction phase; however, this solution would benefit from exact knowledge of the interaction timing from a force- or tactile sensor.

On the development drone, the single most occurring reason for unsuccessful measurements was deviations in relative heading at the moment of impact. While the flexible probe holder helped in correcting the error by getting a larger contact surface, the friction was too low to handle harder impacts. This would cause the drone to slide before the controllers were able to correct. Hence, care should be taken in order to ensure sufficient friction between the probe and the inspection surface. This could for example be achieved using rubber along the edge of the probe holder. However, on the other hardware platform this was not an issue. The supporting frame of the drone helped correct the deviations that might occur, helping the controllers achieve a successful interaction.

The biggest weakness of the current system is the use of local, relative navigation. While the current system is able to carry out the intended mission, global navigation is needed in order to harness the full potential. This improved understanding of the environment would allow the drone to plan and carry out more advanced trajectories, and is the next logical step towards fully autonomous operations.

Chapter 19

Conclusion and Future Work

This chapter presents the main conclusions from this thesis. It also provides suggestions and remarks about future work following the work done in this thesis.

19.1 Conclusions

This thesis gave an overview of several methods for control and navigation in order to do contact based inspections from UAVs. Several modeling approaches have been presented, along with control systems and navigation solutions, to successfully carry out an autonomous thickness measurement.

After thorough testing in simulations and experimental validation, it can be concluded that the impedance based controllers are a robust framework for force control. The position controller has shown great ability to reject noise in the position estimate, and is able to keep the interaction stable for a range of different entrance trajectories. If the interaction requires more explicit control of attitude and velocity than the impedance framework provides, this thesis has presented a solution using pre-generated trajectories, which has been further backed up trough simulations results. This allows for precise tuning of the interaction by constraining attitudes, forces and positions in the trajectory.

The navigation methods developed have through extensive testing and validation shown that they are a robust way of estimating the range and relative orientation to the inspection surface of interest. They can also be extended to work in different regions of the 2D laser scan to estimate the range of obstacles, enabling local, relative navigation. In addition, the navigation methods developed enables implementation of other maneuvers relevant for industrial inspection scenarios, such as a sweeping maneuver following the orientation of the inspection surfaces while the range is locked.

With respect to the safety of the drone platform, environment and operator, the results represent a significant improvement compared to manual flight. During the work on this thesis, no automatic measurements have lead to accidents, while manual flights have experienced several accidents, even with an experienced pilot. Thus, the proposed methods increase the safety of the operation, while also lowering the pilot training necessary to successfully carry out thickness measurements.

This thesis have also showed that the methods developed are hardware agnostic. However, the results also show that care should be taken when designing a hardware platform, in order to best facilitate for contact based inspections. Having a frame that helps minimizing disturbances in yaw angle improves the chances of a valid measurement. A flexible joint should also be included to decrease the sensitivity to the approach angle and better disturbance rejection.

In conclusion, the controllers and navigation methods developed in this thesis enables autonomous thickness measurement. The results showcases complete scenarios with autonomous thickness measurements on multiple hardware platforms. This facilitates safer and more reliable contact based operations, without the need for extensive pilot training.

19.2 Future work

The following will present potential future work following this thesis.

To counteract the problem of set-point manipulations, augmentations to the impedance based controller is possible, such as [31]. Another possible extension to the controller is to use a tactile sensor to turn of the correction of the parallel error during interaction, to avoid destabilizing effects. The hybrid pose/wrench controller presented in [7] is also a possible approach, and future work could include implementation and evaluation of this approach in the simulation framework proposed in this thesis. The method presented in [7] relies on a wrench estimator, and proposes a simple estimation technique based on linear models around steady-state. However, future work can also investigate more advanced methods for wrench estimation, and evaluate how this affects the overall performance of the controller.

This thesis has also outlined an approach to get more detailed control of attitude and velocities during the approach and interaction phase using pre-generated trajectories. The results are promising; however, experimental validations are needed for a decisive conclusion.

Since the controllers developed are based on a passive force control technique, high interaction forces might occur in the transient phase. To counter this problem, an approach using active interaction control can be attempted. This requires the installation of a force sensor in the probe. There are several possible solutions, but a simple approach is to measure the linear deflection of the compliance device and estimate the force based on known spring coefficients. The active force measurement will also better facilitate for a switching controller that uses roll to control yaw during contact.

The Udwadia Kalaba equations is an interesting approach to a detailed simulation of the constrained interaction between the probe and the surface. Future work could include implementation and simulation of the equations derived in this thesis.

Using the relative heading measurements in estimation techniques such as Kalman filtering would allow for fusing with IMU data, potentially providing more accurate estimates and increasing the robustness against sensor outage and

inaccuracies in the RANSAC technique. Tracking of the relative heading over time can also be done using recursive RANSAC [52].

The main drawback of the solutions presented, is that the current navigation solution and sensors are not able to support fully autonomous operations from takeoff to landing. Integrating the control techniques within this thesis in a more advanced navigation framework that provides global positioning information, would improve the performance and capabilities of the solution significantly. One of the more promising methods is using SLAM on data from a 3D scanning LIDAR. Not only would this enable better contact maneuvers, but in combination with path planning algorithms, this allows for advanced fully autonomous operations. The downside of this approach is that most existing 3D LIDARs are quite heavy, and in combination with the weight of other mission payloads, this could result in a bigger and bulkier drone platform that is less suited for indoor inspection.

On the user experience side, integration with camera for accurate position determination would enhance the user experience. An ideal use case is for the operator to click in the video stream in order to indicate where a a thickness measurement should be performed. Since many operations are performed in poorly lit environments, this depends on sufficient light sources on-board the drone. The most predominant drawback of camera based solutions is that the industrial environments often have few features that can be extracted and tracked in an image, which will affect the performance of the navigation algorithms.

Finally, testing with other Non Destructive Testing tools based on contact could also be attempted. For example, a sensor for electrical potential and resistance, which can be used to detect corrosion on steel [9].

Bibliography

[1] Albert Albers et al. “Semi-autonomous flying robot for physical interaction with environment”. In: *2010 IEEE Conference on Robotics, Automation and Mechatronics*. IEEE, June 2010. ISBN: 978-1-4244-6503-3. DOI: 10.1109/RAMECH.2010.5513152. URL: http://ieeexplore.ieee.org/document/5513152/.

[2] Kostas Alexis, Christoph Huerzeler & Roland Siegwart. “Hybrid modeling and control of a coaxial unmanned rotorcraft interacting with its environment through contact”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6631354. URL: http://ieeexplore.ieee.org/document/6631354/.

[3] Kostas Alexis et al. “Aerial robotic contact-based inspection: planning and control”. In: *Autonomous Robots* 40.4 (Apr. 2016). ISSN: 0929-5593. DOI: 10.1007/s10514-015-9485-5. URL: https://doi.org/10.1007/s10514-015-9485-5%20http://link.springer.com/10.1007/s10514-015-9485-5.

[4] J T Bartelds et al. “A comparison of control approaches for aerial manipulators handling physical impacts”. In: *2016 24th Mediterranean Conference on Control and Automation (MED)*. IEEE, June 2016. ISBN: 978-1-4673-8345-5. DOI: 10.1109/MED.2016.7535915. URL: http://ieeexplore.ieee.org/document/7535915/.

[5] A OLLERO BATURONE et al. “The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance”. In: *IEEE Robotics & Automation Magazine* (2018). ISSN: 1070-9932. DOI: 10.1109/MRA.2018.2852789. URL: https://ieeexplore.ieee.org/document/8435987/.

[6] Moisés Jalón Baudet. “Non-linear Attitude Control and Guidance of a Quadrotor UAV”. PhD thesis. 2014.

[7] Steven Bellens, Joris De Schutter & Herman Bruyninckx. “A hybrid pose / wrench control framework for quadrotor helicopters”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, May 2012. ISBN: 978-1-4673-1405-3. DOI: 10.1109/ICRA.2012.6224682. URL: http://ieeexplore.ieee.org/document/6224682/.

[8] A. Bemporad et al. “The explicit solution of model predictive control via multiparametric quadratic programming”. In: *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*. IEEE, 2000. ISBN: 0-7803-5519-9. DOI: 10.1109/ACC.2000.876624. URL: http://ieeexplore.ieee.org/document/876624/.

[9] Luca Bertolini et al. *Corrosion of Steel in Concrete: Prevention, Diagnosis, Repair*. Wiley-VCH, 2014. ISBN: 9783527331468. URL: https://www.amazon.com/Corrosion-Steel-Concrete-Prevention-Diagnosis/dp/3527331468?SubscriptionId=AKIAIOBINVZYXZQZ2U3A%7B%5C&%7Dtag=chimbori05-20%7B%5C&%7DlinkCode=xm2%7B%5C&%7Dcamp=2025%7B%5C&%7Dcreative=165953%7B%5C&%7DcreativeASIN=3527331468.

[10] Marco Bibuli et al. "The MINOAS project: Marine INspection rObotic Assistant System". In: *2011 19th Mediterranean Conference on Control & Automation (MED)*. IEEE, June 2011. ISBN: 978-1-4577-0124-5. DOI: 10.1109/MED.2011.5983104. URL: http://ieeexplore.ieee.org/document/5983104/.

[11] Karen Bodie et al. "An Omnidirectional Aerial Manipulation Platform for Contact-Based Inspection". In: (May 2019). arXiv: 1905.03502. URL: http://arxiv.org/abs/1905.03502.

[12] W Bolton. *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering (3rd Edition)*. Prentice Hall, 2004. ISBN: 0131216333. URL: https://www.amazon.com/Mechatronics-Electronic-Mechanical-Electrical-Engineering/dp/0131216333?SubscriptionId=AKIAIOBINVZYXZQZ2U3A%7B%5C&%7Dtag=chimbori05-20%7B%5C&%7DlinkCode=xm2%7B%5C&%7Dcamp=2025%7B%5C&%7Dcreative=165953%7B%5C&%7DcreativeASIN=0131216333.

[13] Francisco Bonnin-Pascual et al. "A Micro-Aerial platform for vessel visual inspection based on supervised autonomy". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2015-Decem. IEEE, Sept. 2015. ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7353353. URL: http://ieeexplore.ieee.org/document/7353353/.

[14] M. Caccia et al. "MINOAS a Marine INspection rObotic Assistant: system requirements and design". In: *IFAC Proceedings Volumes* 43.16 (Jan. 2010). ISSN: 14746670. DOI: 10.3182/20100906-3-IT-2019.00083. URL: https://www.sciencedirect.com/science/article/pii/S1474667016351035%20http://dx.doi.org/10.3182/20100906-3-IT-2019.00083%20http://linkinghub.elsevier.com/retrieve/pii/S1474667016351035.

[15] M Caris et al. "Synthetic aperture radar at millimeter wavelength for UAV surveillance applications". In: *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE. 2015.

[16] P Cheeseman, R Smith & M Self. "A stochastic map for uncertain spatial relationships". In: *4th International Symposium on Robotic Research*. 1987.

[17] I Can Dikmen, Aydemir Arisoy & Hakan Temeltas. "Attitude control of a quadrotor". In: *2009 4th International Conference on Recent Advances in Space Technologies*. IEEE, June 2009. ISBN: 978-1-4244-3626-2. DOI: 10.1109/RAST.2009.5158286. URL: http://ieeexplore.ieee.org/document/5158286/.

[18] Jonathan Dixon & Oliver Henlich. "Mobile robot navigation". In: *Information Systems Engineering Year, Imperial College* 2 (1997).

[19] Jerry Eaves & Edward Reedy. *Principles of modern radar*. Springer Science & Business Media, 2012.

[20] Olav Egeland & Jan Gravdahl. *Modeling and Simulation for Automatic Control*. 2002.

[21] Markus Eich & Thomas Vogele. "Design and control of a lightweight magnetic climbing robot for vessel inspection". In: *2011 19th Mediterranean Conference on Control & Automation (MED)*. IEEE, June 2011. ISBN: 978-1-4577-0124-5. DOI: 10.1109/MED.2011.5983075. URL: http://ieeexplore.ieee.org/document/5983075/.

[22] Alberto Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989).

[23] Christian Eschmann et al. "Unmanned Aircraft Systems for Remote Building Inspection and Monitoring". In: *European Workshop on Structural Health Monitoring (EWSHM)*. 2012. ISBN: 9783940283412. URL: http://www.ecphm2012.com/Portals/98/BB/th2b1.pdf.

[24] Alexandre Eudes et al. "Autonomous and Safe Inspection of an Industrial Warehouse by a Multi-rotor MAV". In: *Field and Service Robotics*. Springer, Cham, 2018. DOI: 10.1007/978-3-319-67361-5_15. URL: http://link.springer.com/10.1007/978-3-319-67361-5%7B%5C_%7D15.

[25] Rida T. Farouki. *Pythagorean-Hodograph Curves: Algebra and Geometry Inseparable*. Vol. 1. Geometry and Computing. Berlin, Heidelberg: Springer Berlin Heidelberg, Feb. 2008. ISBN: 978-3-540-73397-3. DOI: 10.1007/978-3-540-73398-0. arXiv: 1002.2080. URL: http://link.springer.com/10.1007/978-3-540-73398-0%20http://www.ncbi.nlm.nih.gov/pubmed/22469268%20http://arxiv.org/abs/1002.2080.

[26] Martin A Fischler & Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (June 1981). ISSN: 00010782. DOI: 10.1145/358669.358692. URL: http://doi.acm.org/10.1145/358669.358692%20http://portal.acm.org/citation.cfm?doid=358669.358692.

[27] David A Forsyth. *Computer Vision: A Modern Approach, 2nd Edition*. Prentice Hall, 2011. ISBN: 013608592X.

[28] Bjarne Foss & Tor Aksel N Heirung. *Merging Optimization and Control*. NTNU - Norwegian University of Science and Technology, 2016. ISBN: 978-82-7842-201-4.

[29] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Chichester, UK: John Wiley & Sons, Ltd, Apr. 2011. ISBN: 9781119994138. DOI: 10.1002/9781119994138. URL: http://doi.wiley.com/10.1002/9781119994138.

[30] David Freedman. *Statistical models : theory and practice*. Cambridge New York: Cambridge University Press, 2009. ISBN: 978-0521743853.

[31] Matteo Fumagalli & Raffaella Carloni. "A modified impedance control for physical interaction of UAVs". In: *2013 IEEE/RSJ International Conference on Intelligent*

Robots and Systems. IEEE, Nov. 2013. ISBN: 978-1-4673-6358-7. DOI: 10.1109/IROS.2013.6696619. URL: http://ieeexplore.ieee.org/document/6696619/.

[32] Matteo Fumagalli et al. "Developing an Aerial Manipulator Prototype: Physical Interaction with the Environment". In: *IEEE Robotics & Automation Magazine* 21.3 (Sept. 2014). ISSN: 1070-9932. DOI: 10.1109/MRA.2013.2287454. URL: http://ieeexplore.ieee.org/document/6875943/.

[33] M Fumagalli et al. "Modeling and control of a flying robot for contact inspection". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012. ISBN: 978-1-4673-1736-8. DOI: 10.1109/IROS.2012.6385917. URL: http://ieeexplore.ieee.org/document/6385917/.

[34] Fadri Furrer et al. "Robot Operating System (ROS): The Complete Reference (Volume 1)". In: -. Ed. by Anis Koubaa. Cham: Springer International Publishing, 2016. Chap. RotorS—A. ISBN: 978-3-319-26054-9. DOI: 10.1007/978-3-319-26054-9_23. URL: http://dx.doi.org/10.1007/978-3-319-26054-9%7B%5C_%7D23.

[35] Kenneth Gade. "The Seven Ways to Find Heading". In: *Journal of Navigation* 69.05 (Sept. 2016). ISSN: 0373-4633. DOI: 10.1017/S0373463316000096. URL: http://www.journals.cambridge.org/abstract%7B%5C_%7DS0373463316000096.

[36] C. Galleguillos et al. "Thermographic non-destructive inspection of wind turbine blades using unmanned aerial systems". In: *Plastics, Rubber and Composites* 44.3 (Apr. 2015). ISSN: 1465-8011. DOI: 10.1179/1743289815Y.0000000003. URL: http://www.tandfonline.com/doi/full/10.1179/1743289815Y.0000000003.

[37] Emilio Garcia-Fidalgo et al. "A mosaicing approach for vessel visual inspection using a micro-aerial vehicle". In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Vol. 2015-Decem. IEEE, Sept. 2015. ISBN: 978-1-4799-9994-1. DOI: 10.1109/IROS.2015.7353361. URL: http://ieeexplore.ieee.org/document/7353361/.

[38] Zheng Gong et al. "A compact planar 24GHz quasi-Yagi antenna for unmanned aerial vehicle radar applications". In: *2017 IEEE International Conference on Computational Electromagnetics (ICCEM)*. IEEE. 2017.

[39] Neville Hogan. "Impedance Control: An Approach to Manipulation: Part I - IV". In: *Journal of Dynamic Systems, Measurement, and Control* 107.1 (1985). ISSN: 00220434. DOI: 10.1115/1.3140702. URL: http://dynamicsystems.asmedigitalcollection.asme.org/article.aspx?articleid=1403621%20https://summerschool.stiff-project.org/fileadmin/pdf/Hog1985.pdf.

[40] Peter J Huber. *Robust Statistics (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2003. ISBN: 0471650722.

[41] A.Q.L. Keemink et al. "Mechanical design of a manipulation system for unmanned aerial vehicles". In: *2012 IEEE International Conference on Robotics*

and Automation. IEEE, May 2012. ISBN: 978-1-4673-1405-3. DOI: 10.1109/ICRA.2012.6224749. URL: http://ieeexplore.ieee.org/document/6224749/.

[42] Hassan K Khalil. *Nonlinear Systems*. PEARSON - SUPERPEDIDO, Mar. 2002. ISBN: 0130673897. URL: https://www.xarg.org/ref/a/B00A2KG8B8/.

[43] Kristian Klausen. "Coordinated Control of Multirotors for Suspended Load Transportation and Fixed-Wing Net Recovery". PhD thesis. NTNU, 2017. ISBN: 978-82-326-2417-1. URL: https://brage.bibsys.no/xmlui/handle/11250/2448054?show=full.

[44] N. Koenig & A. Howard. "Design and use paradigms for gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 3. IEEE, 2004. ISBN: 0-7803-8463-6. DOI: 10.1109/IROS.2004.1389727. URL: http://playerstage.sourceforge.net/gazebo/%20http://ieeexplore.ieee.org/document/1389727/.

[45] Taeyoung Lee, Melvin Leok & N. Harris McClamroch. "Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)". In: (Mar. 2010). arXiv: 1003.2005. URL: http://arxiv.org/abs/1003.2005.

[46] Jiaxin Li et al. "Deep learning for 2D scan matching and loop closure". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017.

[47] D. Mader et al. "Potential of UAV-Based laser scanner and multispectral camera data in building inspection". In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. Vol. 2016-Janua. June 2016. ISBN: 16821750 (ISSN). DOI: 10.5194/isprsarchives-XLI-B1-1135-2016. URL: http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLI-B1/1135/2016/%20http://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLI-B1/1135/2016/isprs-archives-XLI-B1-1135-2016.pdf.

[48] Robert Mahony, Vijay Kumar & Peter Corke. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor". In: *IEEE Robotics & Automation Magazine* 19.3 (Sept. 2012). ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2206474. URL: http://ieeexplore.ieee.org/document/6289431/.

[49] Ricardo Martins et al. "IMC: A communication protocol for networked vehicles and sensors". In: *OCEANS '09 IEEE Bremen: Balancing Technology with Future Needs*. IEEE, May 2009. ISBN: 978-1-4244-2522-8. DOI: 10.1109/OCEANSE.2009.5278245. URL: http://ieeexplore.ieee.org/document/5278245/.

[50] N.J. Möller et al. "On the near-wall effects induced by an axial-flow rotor". In: *Renewable Energy* 91 (June 2016). ISSN: 09601481. DOI: 10.1016/j.renene.2016.01.051. URL: https://www.sciencedirect.com/science/article/pii/

S0960148116300519%20https://linkinghub.elsevier.com/retrieve/pii/S0960148116300519.

[51] Michael Montemerlo et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: *Aaai/iaai* 593598 (2002).

[52] Peter C Niedfeldt & Randal W Beard. "Recursive RANSAC: multiple signal estimation with outliers". In: *IFAC Proceedings Volumes* 46.23 (2013).

[53] Jorge Nocedal & Stephen J Wright. *Numerical Optimization*. second. New York, NY, USA: Springer, 2006.

[54] Alberto Ortiz et al. "Vision-Based Corrosion Detection Assisted by a Micro-Aerial Vehicle in a Vessel Inspection Application". In: *Sensors* 16.12 (Dec. 2016). ISSN: 1424-8220. DOI: 10.3390/s16122118. URL: http://www.mdpi.com/1424-8220/16/12/2118.

[55] A Ortiz et al. "First steps towards a roboticized visual inspection system for vessels". In: *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation (ETFA 2010)*. IEEE, Sept. 2010. ISBN: 978-1-4244-6848-5. DOI: 10.1109/ETFA.2010.5641246. URL: http://ieeexplore.ieee.org/document/5641246/.

[56] K.C. Peng et al. "Unmanned Aerial Vehicle for infrastructure inspection with image processing for quantification of measurement and formation of facade map". In: *2017 International Conference on Applied System Innovation (ICASI)*. IEEE, May 2017. ISBN: 978-1-5090-4897-7. DOI: 10.1109/ICASI.2017.7988578. URL: http://ieeexplore.ieee.org/document/7988578/.

[57] Jose Pinto et al. "The LSTS toolchain for networked vehicle systems". In: *OCEANS 2013 MTS/IEEE Bergen: The Challenges of the Northern Dimension*. IEEE, June 2013. ISBN: 9781479900015. DOI: 10.1109/OCEANS-Bergen.2013.6608148. URL: http://ieeexplore.ieee.org/document/6608148/.

[58] José Pinto et al. "Implementation of a control architecture for networked vehicle systems". In: *IFAC Proceedings Volumes (IFAC-PapersOnline)*. Vol. 3. PART 1. Elsevier, Jan. 2012. ISBN: 9783902823199. DOI: 10.1136/oem.2007.034843. URL: https://www.sciencedirect.com/science/article/pii/S1474667016305870?via%7B%5C%%7D3Dihub.

[59] José Pinto et al. "Neptus – A Framework to Support a Mission Life Cycle". In: *7th IFAC Conference on Manoeuvring and Control of Marine Craft*. 2006. URL: https://repositorio-aberto.up.pt/handle/10216/71611%20http://whale.fe.up.pt/Papers/2006/PAPER%7B%5C_%7DMCMC2006-Neptus.pdf.

[60] QGroundControl. *MAVlink Micro Air Vehicule Communication Protocol*. 2017. URL: https://mavlink.io/en/.

[61] Morgan Quigley, Eric Berger & Andrew Y Ng. "STAIR : Hardware and Software Architecture". In: 2007.

[62] Morgan Quigley et al. “ROS : an open-source Robot Operating System”. In: 2009.

[63] Stergios I Roumeliotis, Gaurav S Sukhatme & George A Bekey. “Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization”. In: *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*. Vol. 2. IEEE. 1999.

[64] Björn E. Schäfer et al. “Multicopter unmanned aerial vehicle for automated inspection of wind turbines”. In: *24th Mediterranean Conference on Control and Automation, MED 2016*. IEEE, June 2016. ISBN: 9781467383455. DOI: 10.1109/MED.2016.7536055. URL: http://ieeexplore.ieee.org/document/7536055/.

[65] Jasper L.J. Scholten et al. “Interaction control of an UAV endowed with a manipulator”. In: *2013 IEEE International Conference on Robotics and Automation*. IEEE, May 2013. ISBN: 978-1-4673-5643-5. DOI: 10.1109/ICRA.2013.6631278. URL: http://ieeexplore.ieee.org/document/6631278/.

[66] Martin Stokkeland, Kristian Klausen & Tor A. Johansen. “Autonomous visual navigation of Unmanned Aerial Vehicle for wind turbine inspection”. In: *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, June 2015. ISBN: 978-1-4799-6010-1. DOI: 10.1109/ICUAS.2015.7152389. URL: http://ieeexplore.ieee.org/document/7152389/.

[67] Marco Tognon et al. “A Truly-Redundant Aerial Manipulator System With Application to Push-and-Slide Inspection in Industrial Plants”. In: *IEEE Robotics and Automation Letters* 4.2 (2019).

[68] Alexander J B Trevor, John G Rogers & Henrik I Christensen. “Planar surface SLAM with 3D and 2D sensors”. In: *2012 IEEE International Conference on Robotics and Automation*. IEEE. 2012.

[69] F. E. Udwadia & R. E. Kalaba. “A New Perspective on Constrained Motion”. In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 439.1906 (Nov. 1992). ISSN: 1364-5021. DOI: 10.1098/rspa.1992.0158. URL: http://rspa.royalsocietypublishing.org/cgi/doi/10.1098/rspa.1992.0158.

[70] Luigi Villani & Joris De Schutter. “Force Control”. In: *Springer Handbook of Robotics*. Ed. by Bruno Siciliano & Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. ISBN: 978-3-540-30301-5. DOI: 10.1007/978-3-540-30301-5_8. URL: https://doi.org/10.1007/978-3-540-30301-5%7B%5C_%7D8.

[71] H W Wopereis et al. “Application of substantial and sustained force to vertical surfaces using a quadrotor”. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2017. ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989314. URL: http://ieeexplore.ieee.org/document/7989314/.

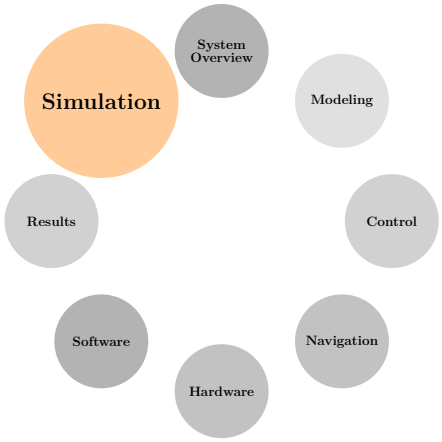
[72] Han W Wopereis et al. "Multimodal Aerial Locomotion: An Approach to Active Tool Handling". In: *IEEE Robotics & Automation Magazine* 25.4 (2018).

[73] D Zhang et al. "Remote inspection of wind turbine blades using UAV with photogrammetry payload". In: *56th Annual British Conference of Non-Destructive Testing - NDT 2017*. Sept. 2017. URL: https://strathprints.strath.ac.uk/63321/.

Appendices

Appendix Part I

Simulation



Appendix A

Background

This chapter will provide a short introduction to the frameworks used in this part, and present the tools and features relevant for this thesis.

A.1 System Overview

The simulation setup consist of three main components that will be presented in the following sections:

- **Gazebo**: Physics simulator
- **Robot Operating System**: Framework for software creation for robots
- **Rotors_simulator**: UAV simulator for Gazebo/ROS.

A.2 Robot Operating System

The development of ROS started in 2007 at Stanford Artificial Laboratory. The project was first called Switchyard, before switching to ROS in 2009 [61, 62]. Contrary to what the name suggests, ROS is not an operating system; however, it provides a collection of tools and libraries to simplify the creation of robots on different platforms. It is based on individual nodes with separate responsibilities, making it flexible and modular. This facilitates for fast prototyping and development of robotic applications.

The three officially supported languages of ROS are C++, Python and Lisp, which can be used interchangeably. ROS only provides official support for Ubuntu (community builds for MacOS exists), and different versions of ROS are specifically built for different version of Ubuntu.

A.2.1 Tools and Features

In ROS, different nodes communicate using a message based system, based on a publish-subscriber architecture. Each node can choose to publish or subscribe to any number of topics, and this also defines the interface of a node. An example of this is a controller that subscribes to a joint state topic and publishes motor references on another topic.

Rviz is a visualization tool for robot state and sensor data, and is bundled together with ROS. It gives the developer access to detailed information and visualization of coordinate frames, links and available sensor data. Also included with ROS is RQT, an alternative GUI based on the QT-framework. RQT provides easy visualization of node communication graphs and tools for debugging data sent on different topics.

In addition, ROS provides tools and libraries for obtaining, building, writing, and running code on multiple computers. This allows for efficient prototyping, development and integration of packages.

A.3 Gazebo

Gazebo is a simulator that was originally developed at the University of Southern California from 2002 [44]. Until 2009 when John Hsu integrated Gazebo into the ROS framework, the main developers were Dr. Andrew Howard and Nate Koening. Gazebo supports different physics engines, but by default it uses the Open Dynamics Engine (ODE)[1]. For rendering the simulation Gazebo uses OGRE[2], giving the user a detailed representation with lighting, shadows and textures. A rendering of a sample simulation can be seen in fig. A.1. Gazebo also allows for remote simulations by separating the simulation (gzserver) and the rendering client (gzclient).

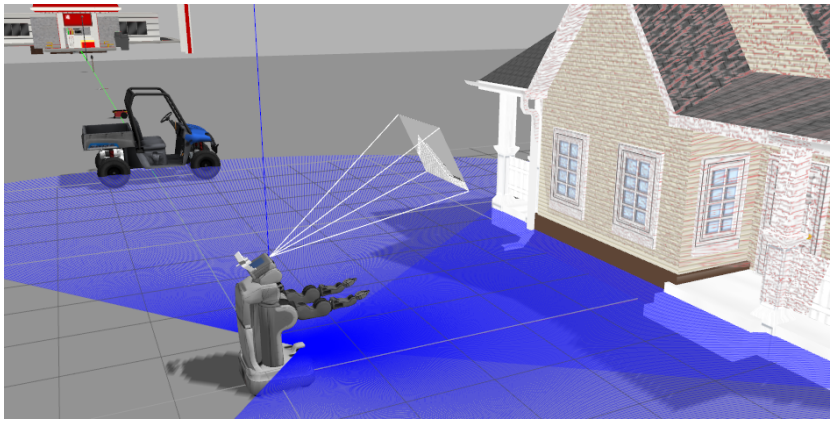


Figure A.1: Example view from Gazebo, also visualizing the robot camera feed and laser sensor data

A.3.1 Tools and Features

Sensors in Gazebo are tightly integrated with the simulator. A base set of high performance sensors is implemented in Gazebo, but the behavior of these can be changed using plugins. For example, a LIDAR sensor can be created by changing

[1] http://www.ode.org/
[2] https://www.ogre3d.org/

the generic laser sensor using C++ code. Altimeter, camera, GPS, IMU and magnetometer are among the available sensors that can be utilized.

Design and editing of environments can either be done by editing SDF files or using the GUI in Gazebo. Model meshes created in other programs such as Blender or AutoCAD can also be imported. Gazebo also comes with a library of models that can be used when creating new environments.

ROS also wraps Gazebo with a tightly coupled interface, and a package called *gazebo_ros* exposes the Gazebo topics in ROS. The connection between ROS and Gazebo is well tested and very stable.

It is important to note the difference between Gazebo and ROS when talking about simulation frameworks. To two comes bundled together, and it is easy to get them mixed up. ROS is a runtime environment and provides tools for software development on robots. Gazebo on the other is a physics simulation tool and provide simulation capabilities for a wide range of mechanical systems and sensors. The tight integration of the two allows for easy testing of the robotic software created in ROS to be tested on mechanical system simulations in Gazebo. The idea is that when the system is ready for hardware testing, the sensor information that previously from Gazebo will come from sensor driver nodes instead. Hence, packages that are developed by the ROS community often come with software modules designed for both ROS and Gazebo, without making a distinct separation between them. Sensor and actuator simulations, as well as 3D models are included for use with Gazebo, while controller frameworks are run in ROS.

A.4 RotorS

ROS has a large and active community, which has created a vast number of packages that can be integrated into personal projects. This easy integration is enabled by the modular architecture of ROS.

Creating a realistic UAV simulator from scratch is time consuming and bug prone. There are a couple of simulators designed for aerial vehicles made available by the community. One of these is the RotorS Simulator [34], which provides simulation capabilities for multirotors. The package was developed by Autonomous Systems Lab (ASL) at ETH, Zürich, a well known institution in the research field of autonomous systems in general and multirotors specifically. The package also

provides simulations of commercially available sensors, such as an IMU, an odometry sensor and the Visual-Inertial Sensor (developed at ASL), which can be mounted on the multirotor.

While simulations always will be a simplification of the real world, the Rotors simulator provides several features to give as realistic simulations as possible. The motors and propellers are simulated individually with a realistic model and a separate package provides thrust simulations based on motor speed and propeller dimensions. The performance of the simulator has been validated experimentally, and at this point it yields realistic simulations to the point where no change in control parameters are needed for the UAVs bundled with the simulator [34]. Figure A.2 shows the AscTec Firefly[3] that is included in the Rotors package.

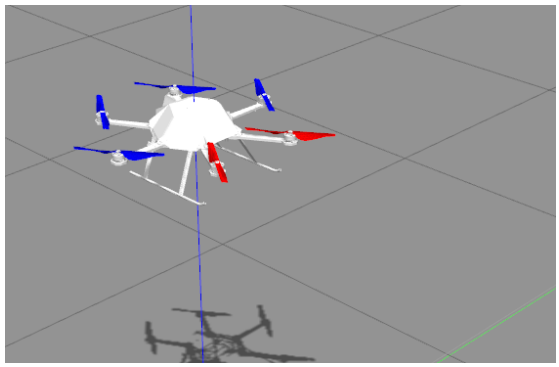


Figure A.2: AscTec Firefly in the Rotors Simulator

[3] http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-firefly/

Appendix B

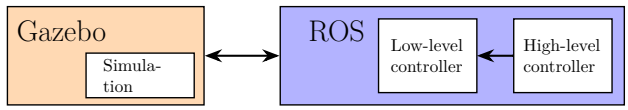
Implementation

This chapter will first present the software framework used to test the controller, and also the setup and interfaces for experimental validation of the controller on the drone.

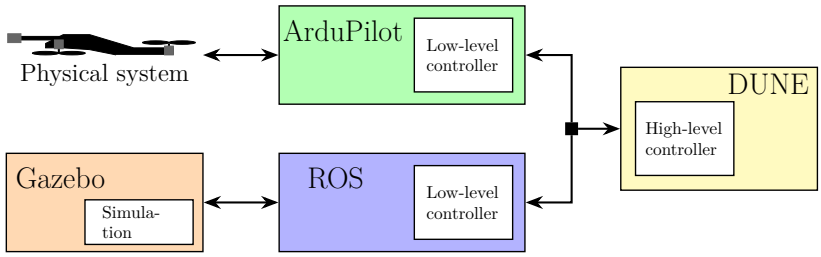
B.1 Simulation Framework Overview

As detailed in part VI, the development drone used in this thesis is running DUNE on its embedded hardware. To facilitate future development, the software framework in this part is constructed to allow controllers with both DUNE and ROS. This gives the opportunity to do fast prototyping and initial testing of controllers in ROS, before moving the controller to DUNE to run it on hardware. Figure B.1 shows the two different scenarios:

- Figure B.1a: Only ROS and Gazebo is used.
- Figure B.1b The interface between ROS and DUNE is the same as between DUNE and the low-level hardware used on the drone, allowing it to both run on the actual hardware and do testing in simulations.



(a) System setup with only ROS and Gazebo



(b) System setup with DUNE

Figure B.1: Different simulation setups

In this thesis all the simulations are done in the framework shown in fig. B.1a. Furthermore, all the interfaces and low-level modules shown in fig. B.1b are implemented and successful control using this interface has been demonstrated. However, work still remains for this to be a stable simulation interface between DUNE and ROS/Gazebo. The key challenges that needs to be solved are ensuring correct timing between the two systems and stabilizing the communication interface. In addition, relevant sensor and actuator needs to be tuned to accurately replicate the behavior of their hardware counterparts.

Figure B.2 shows a more detailed view of how the different modules of the system are interconnected and the interface between them, and presents a more detailed view of the lower branch of fig. B.1b. Note the two different possibilities for placing the controller, shown with gray boxes.

The implementation of the drone using the RotorS Simulator in ROS is described in section B.2, before the creation of the Gazebo environment is presented in section B.3. Finally, the DUNE-ROS interface is detailed in section B.4.

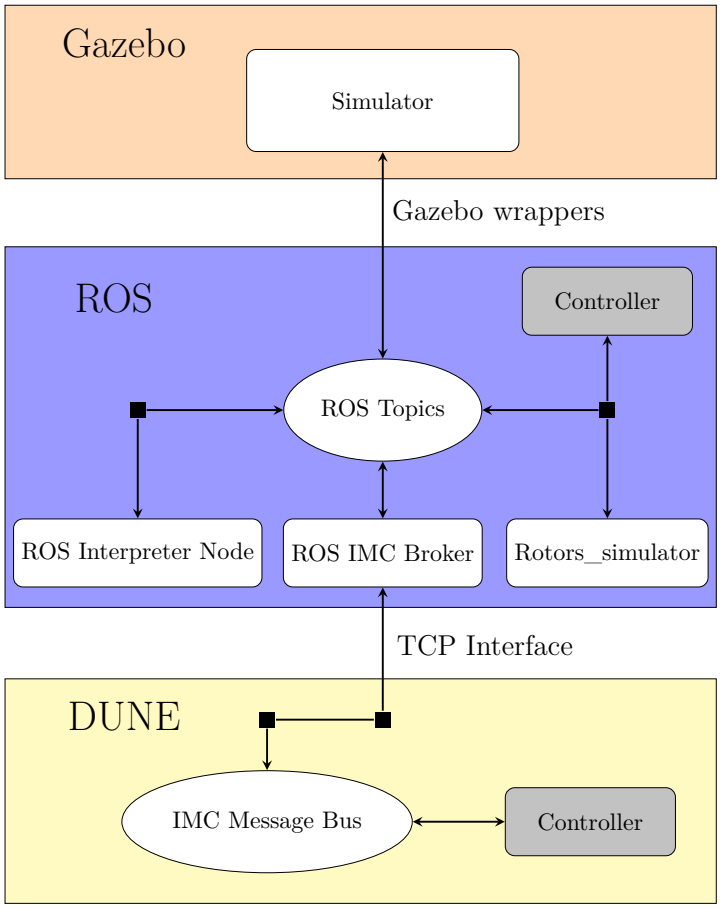


Figure B.2: System overview

B.2 Drone Implementation Using RotorS

Five commonly used drones from the commercial market is bundled together with the *rotors_simulator* package. As described in the previous chapter, sensors can easily be attached to the multirotors to very quickly create simulated sensor data. However, the physical behavior and configuration of the drone is of importance, and hence it was necessary to develop a drone model from scratch using the framework provided by the *rotors_simulator* package. The drone was implemented with the parameters as specified in chapter 2. In addition, the model was augmented with equipment meant to replicate the behavior of the probe attached to the quadrotor, and also a 3D mesh of the drone was used to make the visuals and collisions as realistic as possible. The finished model can be seen in fig. B.3.

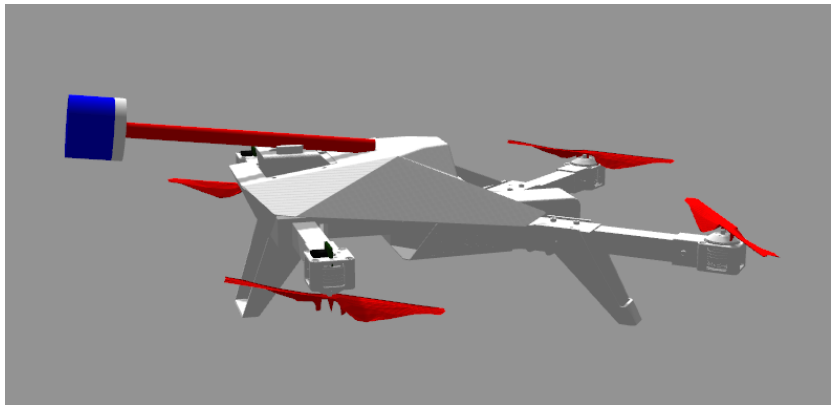


Figure B.3: Visualization of the drone model

The following will give a brief introduction into the necessary parts to recreate the drone in the simulation framework.

B.2.1 URDF Description

There are two ways of representing robots in Gazebo. The first is using the same SDF file used for environment creation to describe the robot model, and the

second is to use the Unified Robot Description Format (URDF), which is an XML format for representing robot models. URDF can be used in combination with Xacro, which is an XML macro language. This makes it possible to create shorter and more readable XML files that expand to larger XML expressions, and for this reason it was chosen as the preferred format.

URDF uses the concept of links and joints to describe a robot. Every robot has a base link that represents the base coordinate system of the robot. Joints are used to connect two links and there exists many different types, such as prismatic, revolute and fixed.

Figure B.4 shows an overview of the entire link and joint setup for the drone that was created.

Low-level attitude controller
The *rotors_simulator* implements a high-gain, low-level attitude controller that can be utilized. The controller is based of "Control of Complex Maneuvers for a Quadrotor UAV using Geometric Methods on SE(3)" [45].

Compliance device
Figure B.3 shows the visualization of the probe and compliance device (blue tip of the probe) that was created. The compliant behavior was simulated using a combination of prismatic joints and soft constraint simulation. The probe is created such that the length, stiffness, placement and angle of the probe can easily be adjusted. Section B.3.2 will provide a detailed proof of the equivalence between soft constraint simulation and a spring-damper compliance system.

Force sensor
The drone has also been equipped with a force-torque sensor (ft_sensor) plugin in the tip of the UTM probe. This is to validate the force exerted by the drone, but also enable testing of control algorithms that utilizes feedback from force in the future.

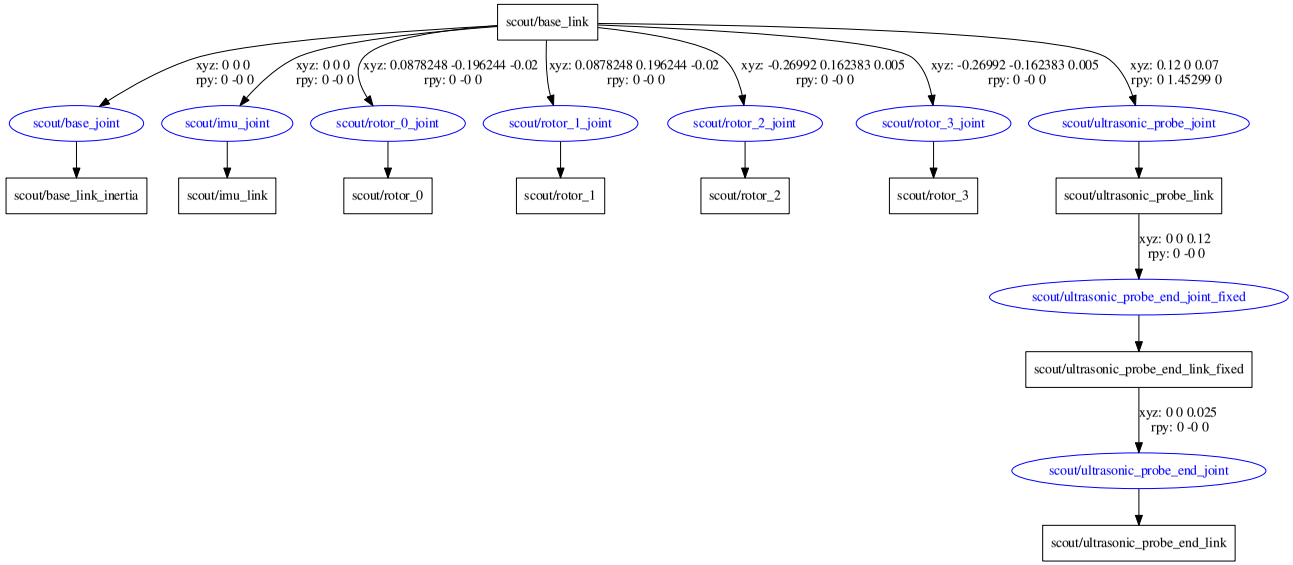


Figure B.4: Overview of the links and joints for the scout robot

B.3 Environment in Gazebo

B.3.1 Creating a World for Interaction

World files used in Gazebo are made using the Simulation Description Format (SDF) file format. SDF is an XML format that describes objects and environments for robot simulators and visualization. It was originally developed as part of Gazebo, and was designed with special emphasis on scientific robot applications. SDF is a complex file format, and this section will only provide a short introduction to the parts essential to this thesis. For any further details, please refer to the official documentation[1].

Global Physics Engine Properties

The SDF file needs to specify the physics engine to be used, and parameters for the selected engine. To allow for accurate real-time simulations, the ODE physics engine was used with a max step size of 0.01 s and the solver type was set to *quick* with a maximum of 1000 iterations.

Friction

By default the Open Dynamics Engine (ODE) physics engine is setup to use very little friction between elements. For the interaction considered in this thesis the result depends significantly on correct friction parameters, so the friction had to be changed to better imitate the real world. In listing B.1 the essentials for changing the friction of a model is shown. *mu* and *mu2* are the friction coefficients used by the physics engine. The *fdir1* parameter is the direction to calculate the friction in the local collision frame, and hence the magnitude of the vector is ignored. With the vector defined as in listing B.1, the friction forces will be parallel to the surface of the interaction.

In addition, the collision surface model is declared static, such that it becomes immovable. This is fitting to the physical interpretation of a wall, but also

[1] http://sdformat.org/spec

significantly reduces the calculation load on the physics engine during interaction. As only one of the objects in the collision are dynamic, simplifications can be made without loss of precision.

```
<model name="wall">
      ...
      <surface>
        <friction>
          <ode>
            <mu>1</mu>
            <mu2>1</mu2>
            <fdir1>1.0 1.0 0</fdir1>
          </ode>
        </friction>
      </surface>
      ...
</model>
```

Listing B.1: Specifying friction parameters for the wall

B.3.2 Simulating Interaction in Gazebo and ROS

The following section presents a proof equivalence between simulating a mass-spring-damper system using implicit Euler and error correction with soft constraints using semi-implicit Euler. This is an essential part in how to implement the compliance in the UTM probe when interacting with the environment in a stable manner.

Simulating springs is in general not a trivial task. The spring stiffness heavily influences the simulation and it can even become unstable, depending on the choice of integrator [20].

Equations (B.1) to (B.3) shows the different integration schemes for the explicit, implicit and semi-implicit Euler methods for a one-dimensional system with position and velocity (Differences are highlighted in red).

$$\begin{aligned} &\text{Explicit Euler} \\ &x_{t+1} = x_t + h\textcolor{red}{v_t} \\ &v_{t+1} = v_t + h\frac{\partial v}{\partial t}\bigg|_{\textcolor{red}{x=x_t, v=v_t}} \end{aligned} \tag{B.1}$$

Implicit Euler

$$\begin{aligned} x_{t+1} &= x_t + h v_{t+1} \\ v_{t+1} &= v_t + h \frac{\partial v}{\partial t}\bigg|_{x=x_{t+1}, v=v_{t+1}} \end{aligned} \tag{B.2}$$

Semi-implicit Euler

$$\begin{aligned} x_{t+1} &= x_t + h v_{t+1} \\ v_{t+1} &= v_t + h \frac{\partial v}{\partial t}\bigg|_{x=x_t, v=v_t} \end{aligned} \tag{B.3}$$

As seen in eq. (B.2), the implicit Euler integrator uses the position and velocity from the next time step in the update step. This means that an implicit equation set has to be solved for each update step. While the equations above are fairly easy to solve, it constitutes a massive increase in computations needed. This is unsuitable for real-time applications, despite the fact that it provides the best properties for stable simulation.

As mentioned in section B.3, Gazebo uses the Open Dynamics Engine (ODE) as its physics engine. As most other physics engines, it uses semi-implicit Euler as integration schema. This is a compromise between the speed of explicit Euler and robustness of implicit Euler. As shown by [20], the stable simulation of springs without using implicit Euler is challenging, as the stability depends on the stiffness of the system and the integration time h. Hence most physic engines employ another strategy called soft constraints to deal with this issue. This relieves the developer of the task of choosing the tricky relation between spring constants and simulation step-length to ensure numerical stability.

The equations for soft constraints used by ODE is given as follows:

$$m \frac{\partial v}{\partial t} = \lambda \tag{B.4}$$

$$v + \frac{\beta}{h} x + \gamma \lambda = 0 \tag{B.5}$$

Inserting this system into the semi-implicit Euler integration scheme yields

$$x_{t+1} = x_t + hv_{t+1} \tag{B.6}$$

$$v_{t+1} = v_t + h\frac{\lambda}{m} \tag{B.7}$$

$$v_{t+1} + \frac{\beta}{h}x_t + \gamma\lambda = 0 \tag{B.8}$$

Combining eq. (B.7) and eq. (B.8) and solving for v_{t+1} yields:

$$v_{t+1} = \frac{v_t - \frac{\beta}{m\gamma}x_t}{1 + \frac{h}{m\gamma}} \tag{B.9}$$

To compare the method of soft constraints, a classic 1D mass-spring-damper system is first defined:

$$\frac{\partial x}{\partial t} = v \tag{B.10}$$

$$m\frac{\partial v}{\partial t} = -cv - kx \tag{B.11}$$

Where k and c are the spring and damper constants respectively. Solving this system using the implicit Euler method yields

$$x_{t+1} = x_t + hv_{t+1} \tag{B.12}$$

$$v_{t+1} = v_t - \frac{h}{m}(cv_{t+1} + kx_{t+1}) \tag{B.13}$$

Again, solving for v_{t+1} yields

$$v_{t+1} = \frac{v_t - \frac{hk}{m}x_t}{1 + \frac{hc}{m} + \frac{h^2k}{m}} \tag{B.14}$$

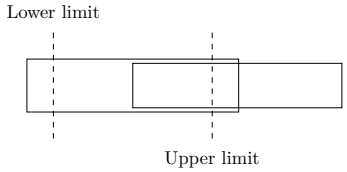
Comparing the solutions from soft constraints with semi-implicit Euler to the solution in eq. (B.14), it becomes apparent that setting

$$\begin{aligned} \gamma &= \frac{1}{c + hk} \\ \beta &= \frac{hk}{c + hk} \end{aligned} \tag{B.15}$$

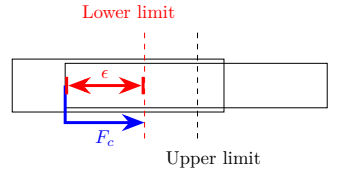
yields the same solution.

This shows the equivalence between simulating a mass-spring-damper system with implicit Euler using spring and damper constants k and c, and using soft constraints in combination with semi-implicit Euler and γ and β defined as in eq. (B.15).

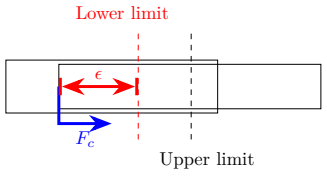
The implementation specifics of a spring-damper compliance device is shown in fig. B.5. Figure B.5a shows a general prismatic joint with its upper and lower limits. In case of a violation of the constraint, the ODE physics engine will try to correct this in one integration step, as illustrated in figure fig. B.5b. However, changing γ and β alters this behavior, and will cause the violation to be corrected over multiple time steps, as depicted in fig. B.5c. By adjusting the lower and upper limits to the same place, and choosing γ and β based on eq. (B.15) and the desired k and c. This will give compliance according to a spring damper system, as illustrated in figure fig. B.5d.



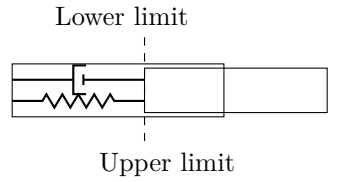
(a) Prismatic joint with limits



(b) Prismatic joint with constraint violation, correction in one time step



(c) Prismatic joint with constraint violation, correction in multiple time steps



(d) Limits are equal, forces equivalent with spring-damper system

Figure B.5: Implementation steps of spring-damper compliance device

B.4 DUNE-ROS Interface

The interconnection between DUNE and ROS is facilitated by an IMC broker node in ROS, which connects to the IMC bus via a TCP interface. The Broker node was created by OceanScan and is available online[2]. The IMC messages are wrapped into a ROS message and sent to the Interpreter Node. Since most of the message types sent in ROS are not compatible with the messages sent in DUNE, it is necessary to translate them into the correct message format. This is done by the interpreter node, and needs to be programmed to convert the correct messages and publish them to the correct topics. A visualization of the communication can be seen in fig. B.2. Below is a list of all the message type conversions that is done in order to get compatible types:

Messages from ROS to DUNE

- nav_msgs::Odometry → IMC::NedState
- sensor_msgs::Imu → IMC::Acceleration
- sensor_msgs::Imu → IMC::AngularVelocity

Messages from DUNE to ROS

- IMC::DesiredCopterControl → mav_msgs::RollPitchYawrateThrust

An important note is that ROS and Gazebo uses the NWU convention for coordinate systems while DUNE uses NED. This convention difference is also carried over to direction of the axis for the body frame, such that the x-axis is pointing forward in both DUNE and ROS/Gazebo while the z-axis is pointing up in Gazebo and ROS and down in DUNE. The y-axis completes the right hand rule in both cases. This makes it necessary to convert the values sent in the messages, such that they are represented in the appropriate coordinate system.

[2] https://github.com/oceanscan/ros-imc-broker

Appendix C

Simulation Results

This chapter will present the result of different simulated scenarios using ROS and the Gazebo physics engine. For three of the simulations, a video is also provided. These can be accessed by using the QR-code or URL provided.

Note that the simulations have been carried out using ROS/Gazebo, even though the visualizations are created in MATLAB. All data from the simulations were recorded and imported to MATLAB in order to create more intuitive visualizations than what is provided in ROS. However, the drone visualization in MATLAB is simplified, so small deviations between the visualizations and the simulated environment might occur.

The inspection surface was placed at $x_w = -2\,\text{m}$ parallel with the y-axis. The UAV starts at $\mathbf{p}_b^n = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top$ facing towards positive x. The starting frame of the simulations can be seen in fig. C.1. When the drone takes off, it rotates 180° before approaching the inspection surface. The take-off and first part of the rotation is not included, which is why the time index starts at $t = 5\,\text{s}$ in the following simulations.

Since the inspection surface is parallel with the y-direction, the terms ”parallel to the inspection surface” and ”perpendicular to the inspection surface” is used interchangeably with ”y-direction” and ”x-direction” respectively in the following. This is to simplify the wording in the discussions, and is of no other significance to the results. The word ”angle-of-attack” is also used to describe the angle between the vector along the probe and the vector perpendicular to the inspection surface at impact, for the same reason of simplification.

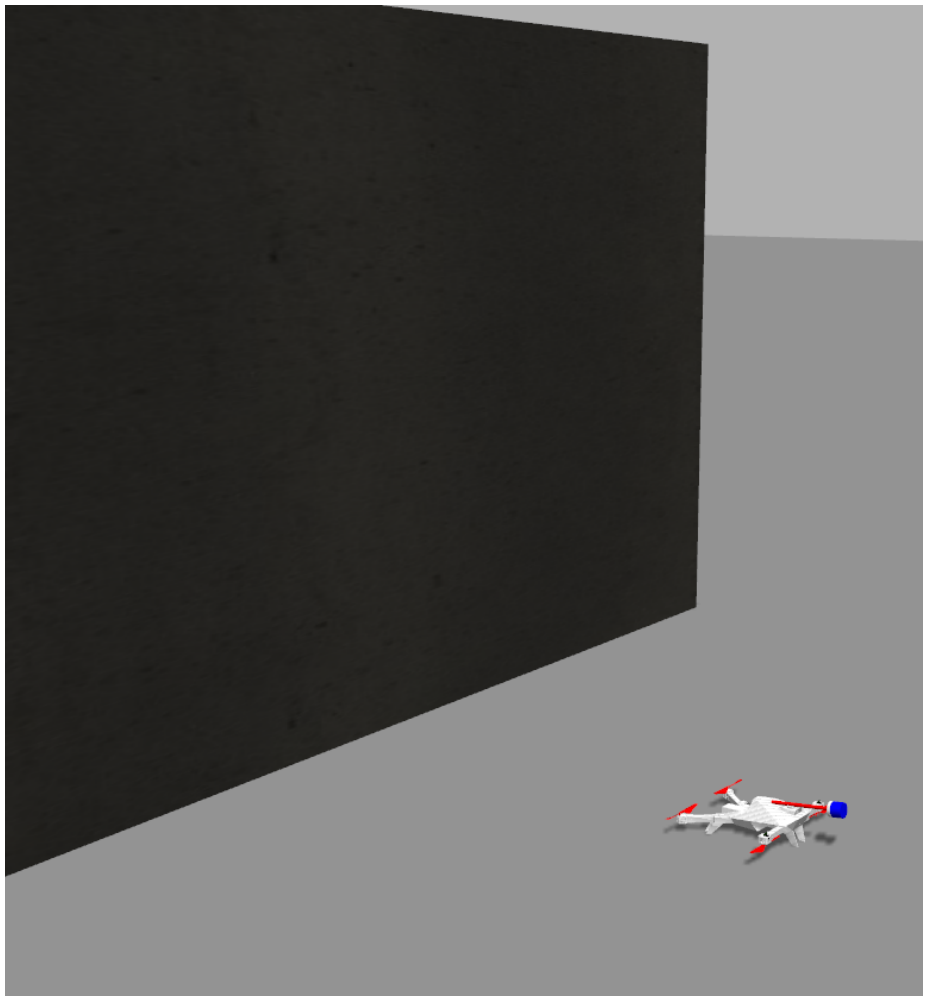


Figure C.1: Screenshot of the initial state in the simulations

The scenarios that will be considered are:

- **Scenario I** - *Perfect position measurements and straight approach*
  This scenario will function as a benchmark for the rest of the simulations. (Video available)
- **Scenario II** - *Noisy position measurements*
  Perfect knowledge about the position to the surface of interaction is unrealistic and these scenarios aims to investigate the influence of disturbances in the position estimate. Uniform noise is added to the true position given by the odometry sensor. The following simulations are conducted for this scenario:
  - **II.1**: Uniform noise with spread of ± 5 cm in the x-direction. (Video available)
  - **II.2**: Uniform noise with spread of ± 25 cm in the x-direction.
  - **II.3**: Uniform noise with spread of ± 5 cm in the x-direction and ± 10 cm in the y-direction.
- **Scenario III** - *Varying angle-of-attack*
  These scenarios aim to investigate the response of the system when the vector of the probe is not perpendicular to the vector along the inspection surface. The following simulations are carried out for this scenario:
  - **III.1**: 5° angle-of-attack.
  - **III.2**: 10° angle-of-attack. (Video available)
  - **III.3**: 10° angle-of-attack and minimal friction.
- **Scenario IV** - *Different spring and probe configurations*
  In these scenarios different configurations are considered to see how they affect the stability and robustness of the overall interaction. The following simulations are conducted for this scenario:
  - **IV.1**: Compliance device is changed to a long spring with lower spring stiffness
  - **IV.2**: Different configuration and placement of the probe.

The sections below will present the main findings from each simulation, before section C.5 will summarize and discuss the overall results.

C.1 Scenario I

In this scenario the drone has perfect position and attitude information. The simulations clearly shows that the drone is able to stabilize in contact with the surface of the wall. After slight bouncing in the initial phase of the interaction due to the compliance device, the contact forces are stable at approximately 3.75 N. This is well above the target at 3 N. The contact induces small deviations of about 3° in the yaw angle; however, they are kept constant at this small angle by the controller. In fig. C.2 it can be seen that the pitch controller stabilizes around 6°. The steady-state of the interaction can be seen in fig. C.3.

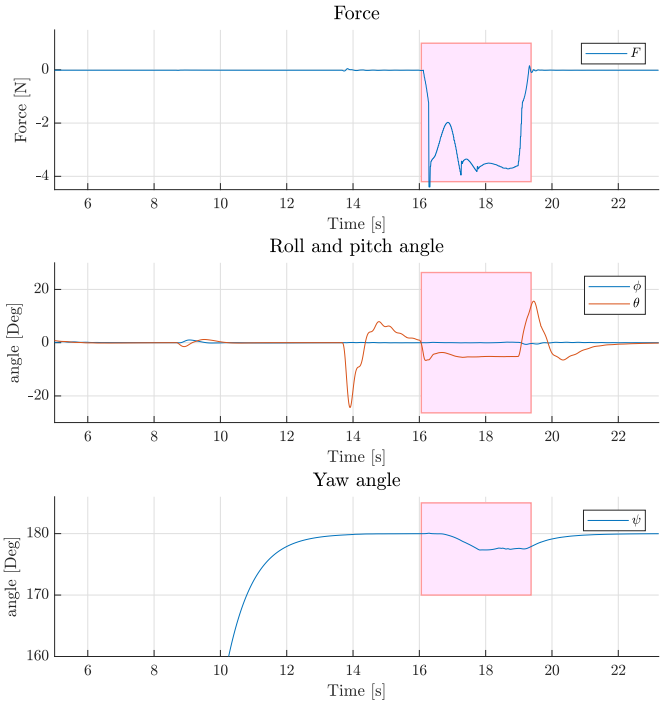


Figure C.2: Attitude and force for baseline simulation. (Red area shows the duration of contact)

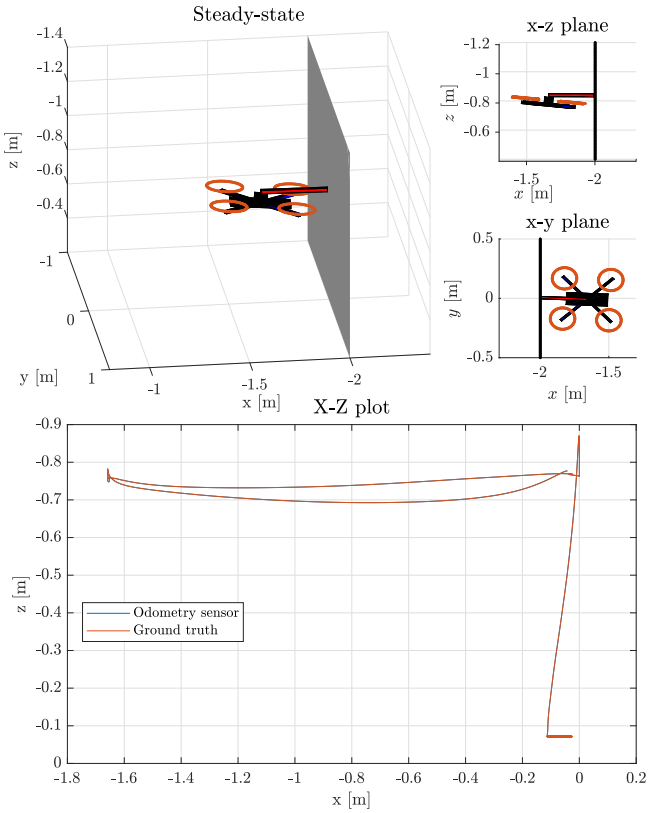


Figure C.3: Baseline simulation.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

Figure C.4: QR-code for the above simulation (youtu.be/HdgM44mmjDQ)

C.2 Scenario II

In this scenario noise is added to the odometer measurements. Different noise levels are simulated, two with noise only in the direction of the interaction and one with noise in both x- and y-direction.

C.2.1 Simulation II.1

First, the effect of an additive uniform noise with a spread of 5 cm in the x-direction is simulated. Figures C.5 and C.6 shows the results of the simulations. Once again, the results clearly shows that the drone is able to stabilize in contact with the surface of the wall and the results are very similar to the baseline results. However, the steady-state force is slightly lower, but still over the targeted 3 N.

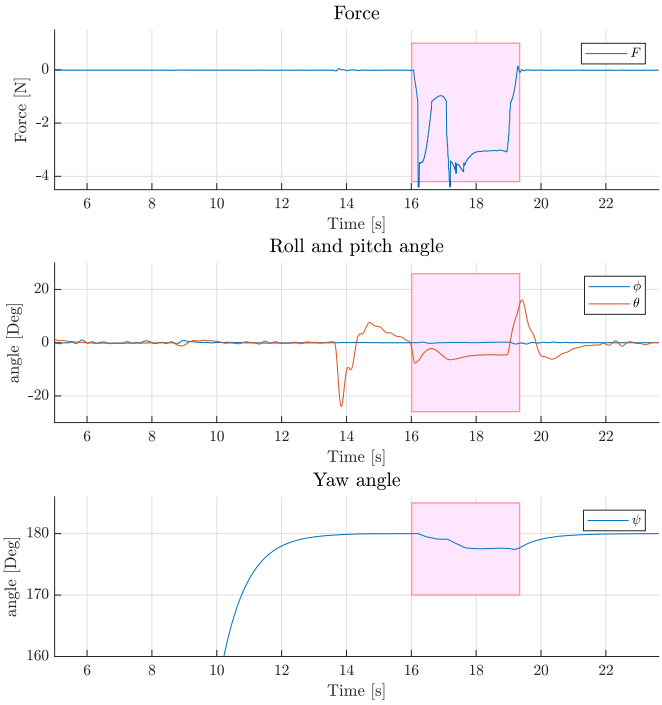


Figure C.5: Attitude and force for simulation with spread of 5 cm. (Red area shows the duration of contact)

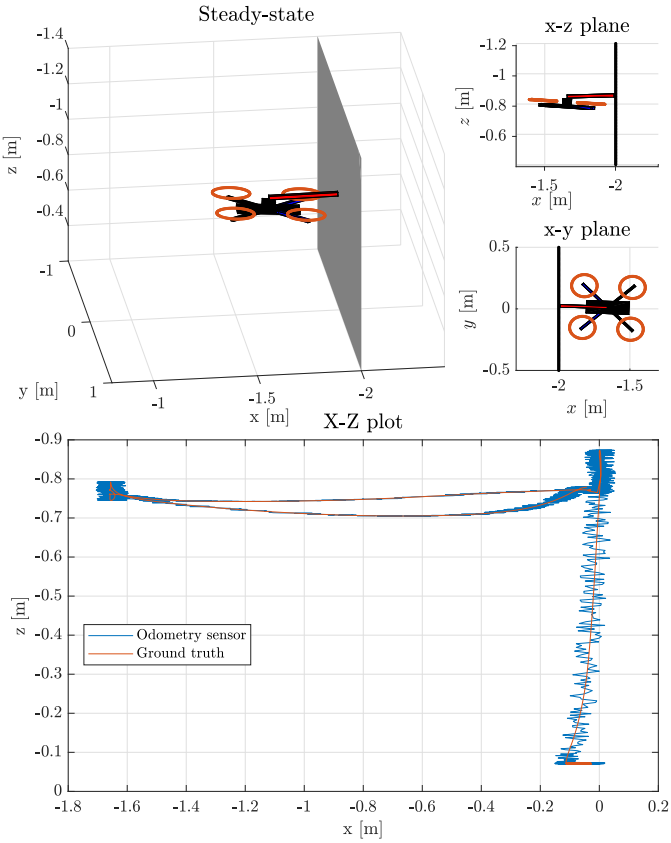


Figure C.6: Simulation with x-spread of 5 cm.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

Figure C.7: QR-code for the above simulation (youtu.be/Hwi8KqHY9BU)

C.2.2 Simulation II.2

In the second simulation, the effect of a much larger noise with a spread of 25 cm is simulated. Figures C.8 and C.9 below shows the results of the simulations. Despite the massive variations in the position given to the controller, the drone is able to stabilize in contact with the inspection surface. This shows the robustness of the controller, as there are only small differences between the results in this simulation and the previous with less noise.

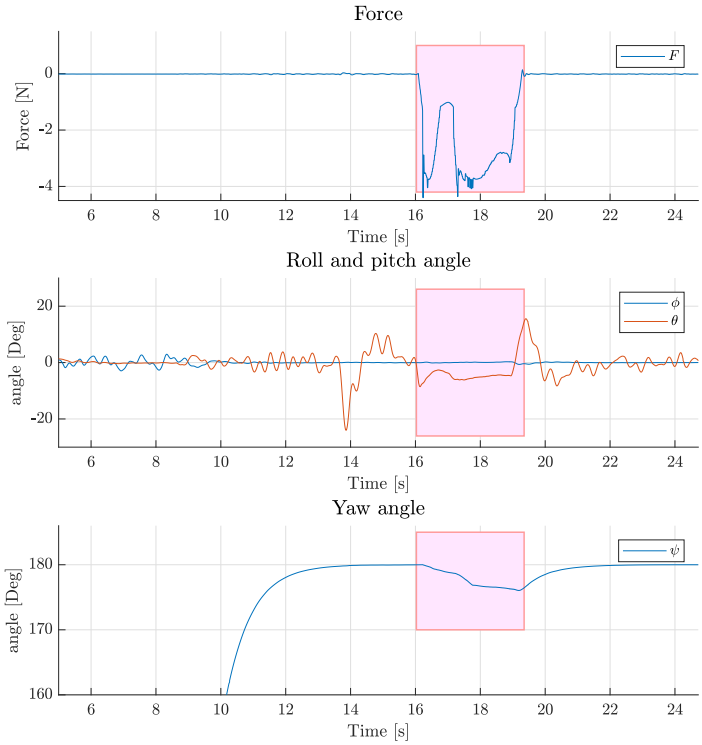


Figure C.8: Attitude and force for simulation with spread of 25 cm. (Red area shows the duration of contact)

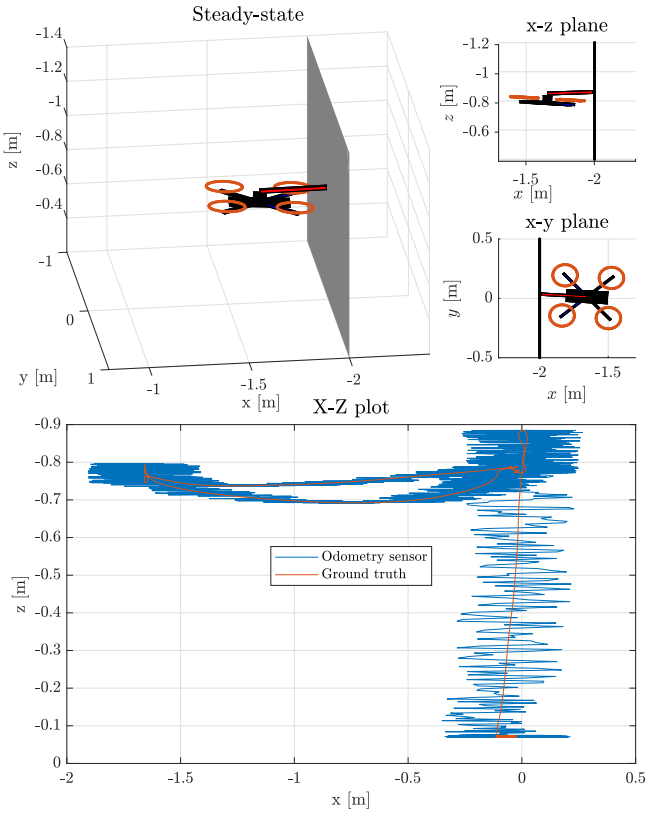


Figure C.9: Position simulation with spread of 25 cm.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

C.2.3 Simulation II.3

Lastly, noise is also added to the direction perpendicular to the inspection surface. In figs. C.10 and C.11 the simulations with spread of 15 cm in y-direction and 5 cm in x-direction and is shown. While the interaction is still stable, the contact force is now oscillating around 3 N. Also, the yaw angle is drifting during this interaction, and while it stabilizes towards the end off the interaction, the propellers are getting very close to the surface. The reason for this is that the controller is trying to correct the y position during the interaction; however, this is not possible due to the constrained trajectory. It might be beneficial to explicitly tell the controller not to correct the y position during interaction, since attempts at sideways control during interaction will only destabilize the interaction. However, this does introduce the need to accurately determine when the UAV is interaction with the environment, possibly using a force- or tactile sensor. Simulations carried out with a lower spread of 10 cm in the y-direction was successful, so despite the results discussed above, the controller shows good disturbance rejection capabilities also in this direction.

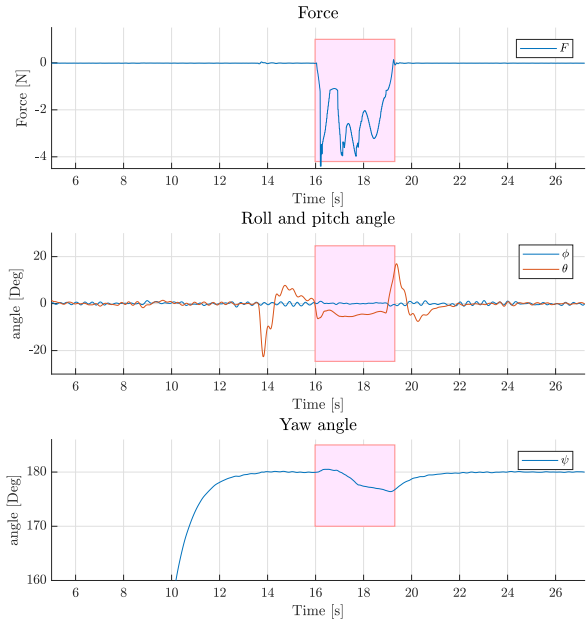


Figure C.10: Attitude and force for simulation with spread of 5 cm in the x-direction and 15 cm in the y-direction. (Red area shows the duration of contact)

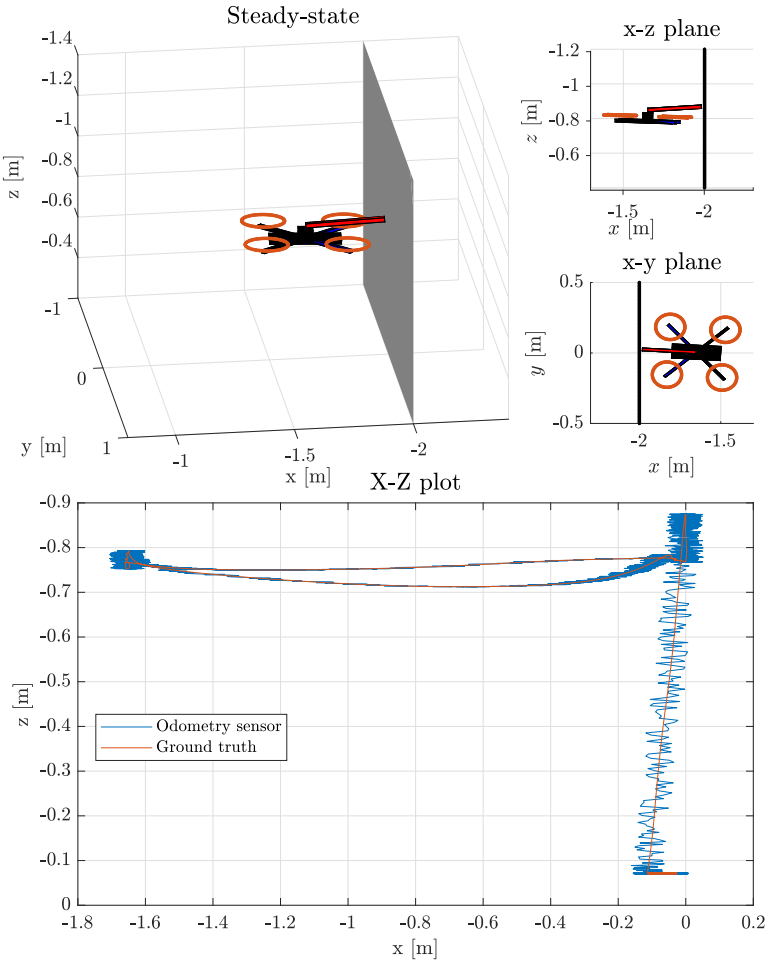


Figure C.11: Spread of 5 cm (x-direction) and 15 cm (y-direction).
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

C.3 Scenario III

This scenario aims to simulate the effect of a angled impact with the inspection surface. This is simulated by adjusting the yaw angle of the approach.

C.3.1 Simulation III.1

First, the yaw angle is set to 5° during the impact. The results are given in figs. C.12 and C.13, and shows that the controller are able to correct the angled approach. However, the yaw angle is not perfectly corrected, but stabilizes around the same value as in the baseline simulation. The contact force is also stable above 3 N, and the simulation shows a successful docking.

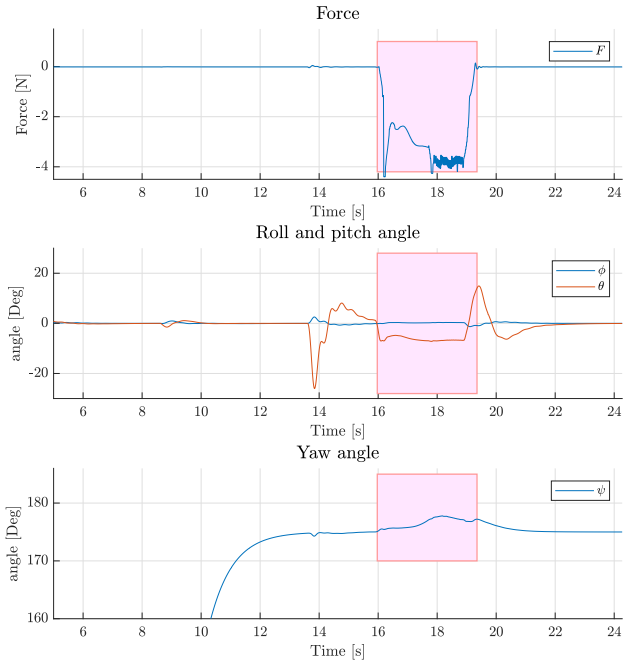


Figure C.12: Attitude and force for simulation with an angle-of-attack of 5°. (Red area shows the duration of contact)

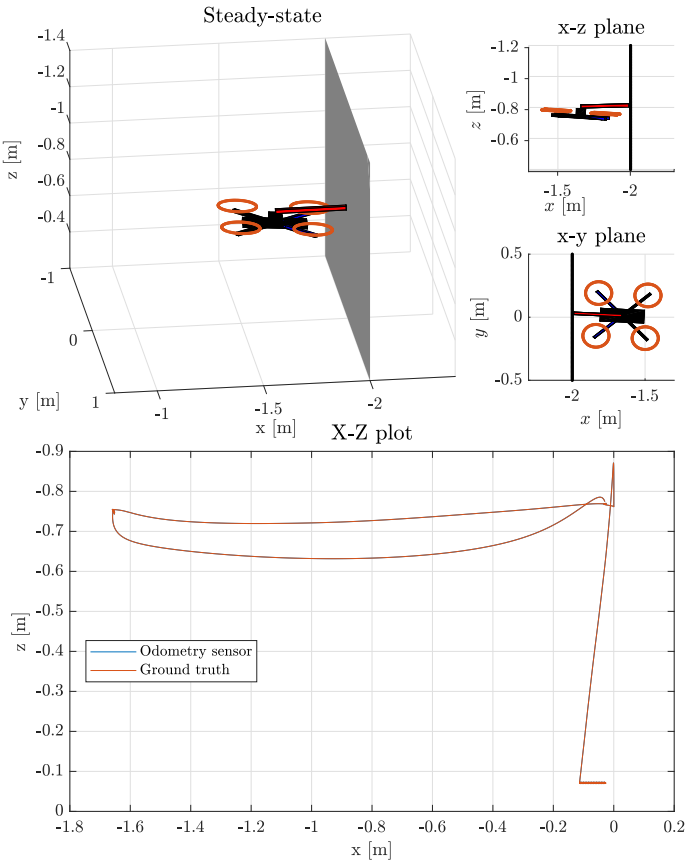


Figure C.13: Simulation with an angle-of-attack of 5°.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

C.3.2 Simulation III.2

The following simulation also show that the controller can handle interaction with an even larger angle-of-attack. The yaw angle was set to 10° during the impact, and the results are given in figs. C.14 and C.15. The contact force is once again stable above 3 N, and even larger than in the baseline interaction. This is because the forces used to correct the interaction also adds to the resulting measurement. Also here, the yaw angle is corrected similar to the simulation in section C.3.1. The noise in the force sensor is a result of the sampling technique used in the sensor, and does not originate from an unstable interaction.

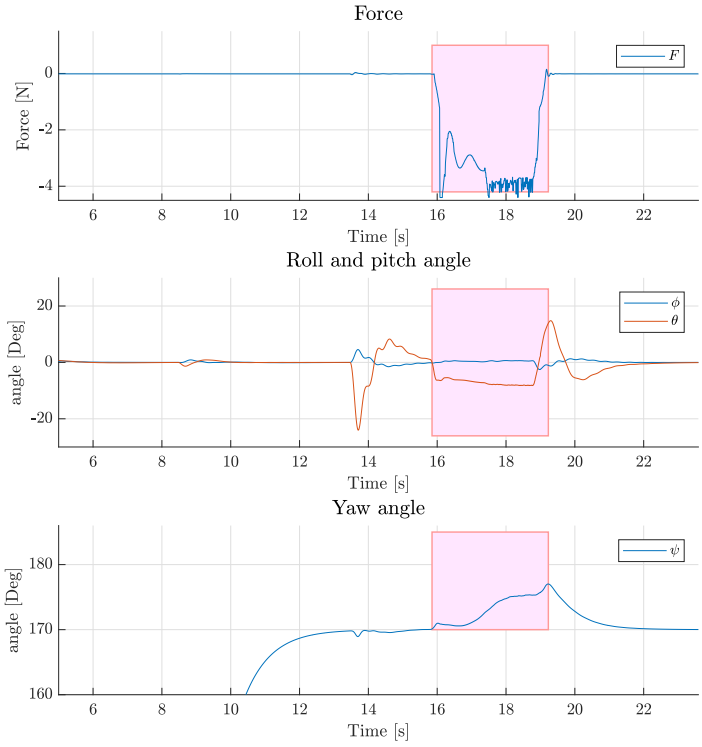


Figure C.14: Attitude and force for simulation with an angle-of-attack of 10°. (Red area shows the duration of contact)

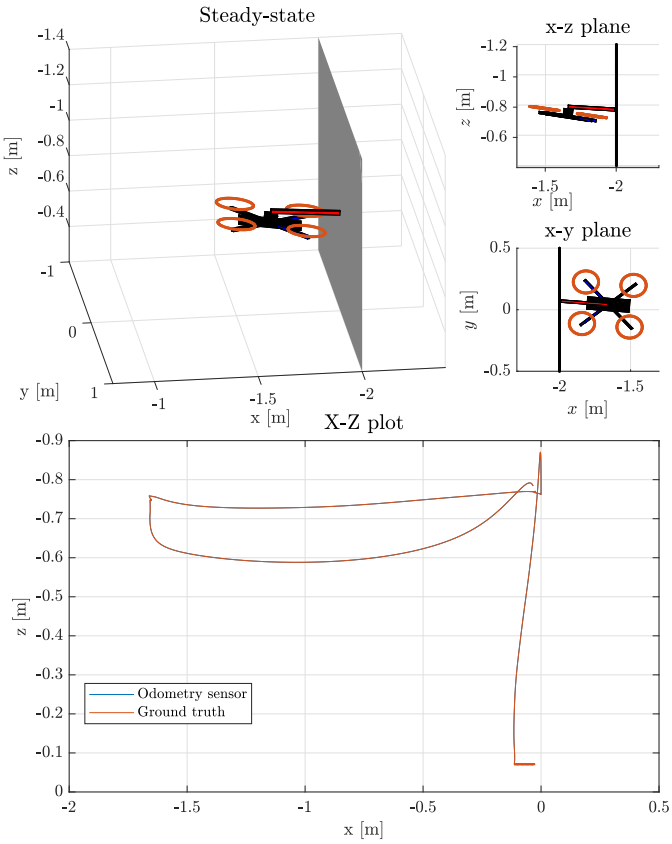


Figure C.15: Simulation with an angle-of-attack of 10°.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

Figure C.16: QR-code for the above simulation (youtu.be/W_J9lE4-t0k)

C.3.3 Simulation III.3

However, when the probe hits the surface at an angle, the importance of friction becomes apparent. In the simulations shown in figs. C.17 and C.18 the friction is set to a minimum, and as a result the drone crashes. When the drone hits the surface with an angle of 10°, the impact causes the probe to slide and the rotors comes into contact with the inspection surface. The best way to counter this problem, is to design the hardware with friction in mind. Padding the compliance device with a rough/rubber material is a potential solution, which give larger friction forces to facilitate a more stable interaction.

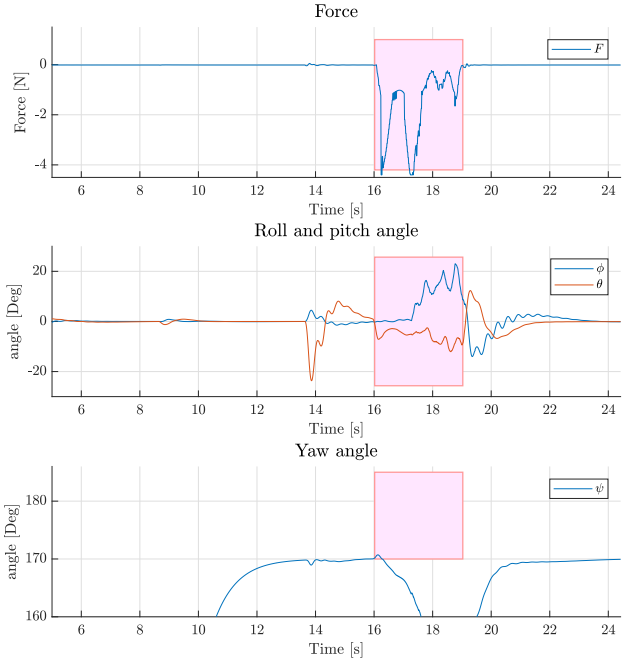


Figure C.17: Attitude and force for simulation with an angle-of-attack of 10° and low friction. (Red area shows the duration of contact)

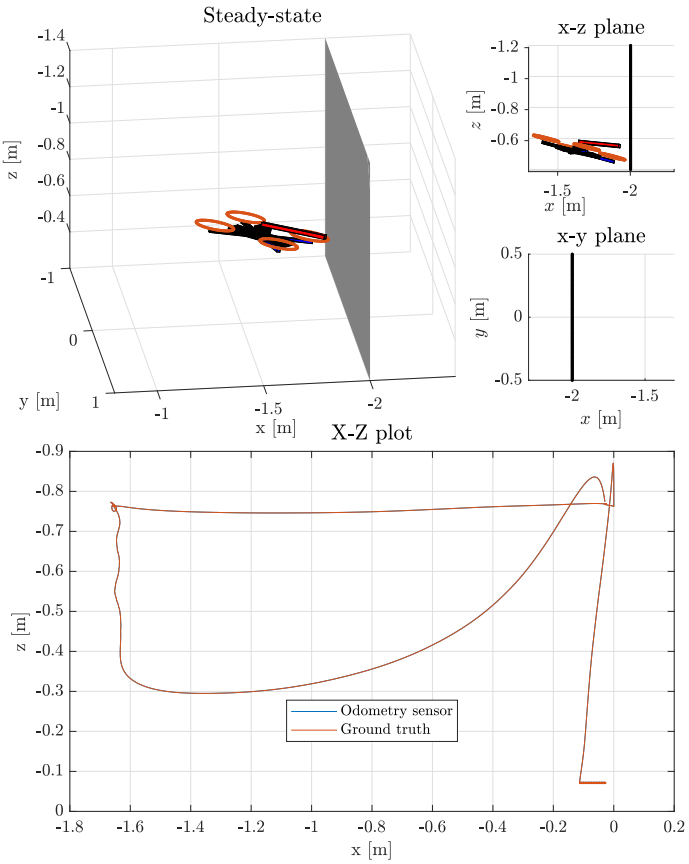


Figure C.18: Simulation with an angle-of-attack of 10° and low friction.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

C.4 Scenario IV

In this section the aim is to look at how different configurations of the probe/-compliance device affects the simulation results.

C.4.1 Simulation IV.1

The first simulation in this scenario tests the effect of using a long spring with lower stiffness, to give a ”softer” initial contact. The results are shown in figs. C.18 and C.19. Also in this experiment the drone was approach the surface at an angle, and interestingly the soft spring is much better to correct the yaw angle than it than its stiffer counterpart. This is mainly due to fact that the softer spring helps to guide the drone towards a more straight trajectory. Also, the slight bouncing behavior shown in the beginning of the previous simulations (for example in fig. C.2) are eliminated using this approach, due to the softer spring. However, the UAV loses more altitude during this interaction, as the drone is pushing slightly downwards. This is because the rigid part of the probe is shorter, giving the contact forces less arm to create moments about, and hence the same moments from the drone creates a larger pitch angle. Careful hardware design and tuning the controller specifically for this scenario would help to counter this problem.

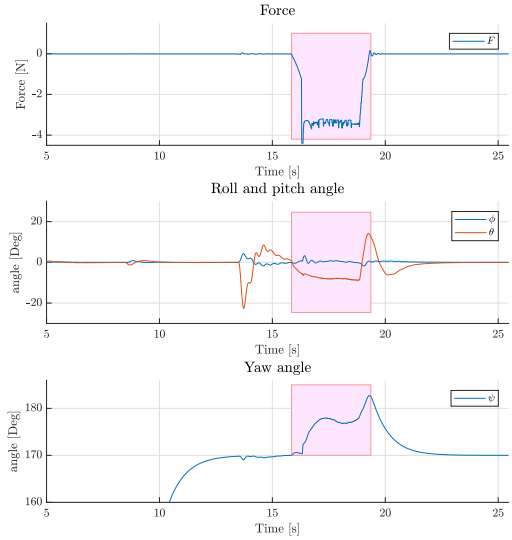


Figure C.19: Attitude and force for simulation long spring and angle-of-attack of 10°. (Red area shows the duration of contact)

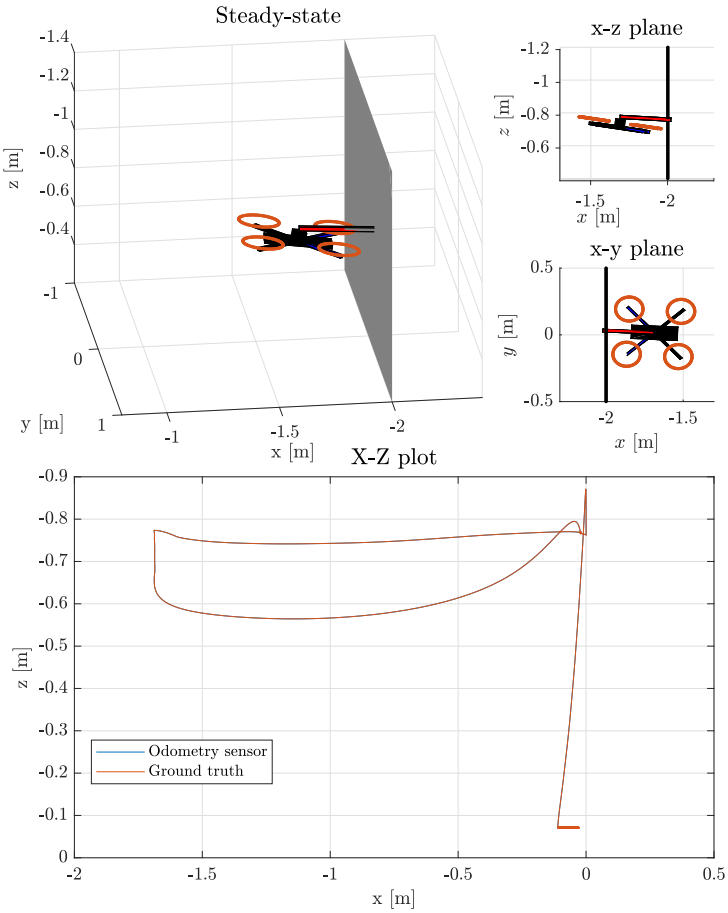


Figure C.20: Simulation with a long spring and angle-of-attack of 10°.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

C.4.2 Simulation IV.2

In the second trial, a different configuration of the probe was attempted. Instead of attaching the probe in front of the drone, it was attached slightly higher, with anchoring in the back part of the UAV (see fig. C.22 for a visualization). The idea is that such a configuration will free up space in front of the drone for sensors or other mission payloads. The results are given in figs. C.21 and C.22, and show no significant differences compared to the previous simulations. The drone produces slightly more force against the inspection surface; however, the yaw angle also drifts slightly more than in the baseline simulation. In conclusion, this configuration is a totally viable option compared to the traditional front mounting.

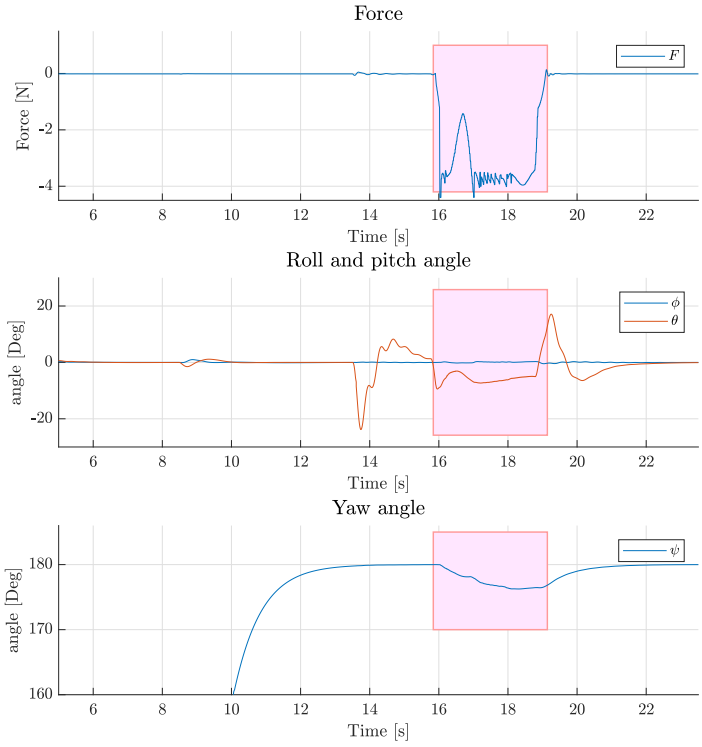


Figure C.21: Attitude and force with alternative configuration of the drone. (Red area shows the duration of contact)

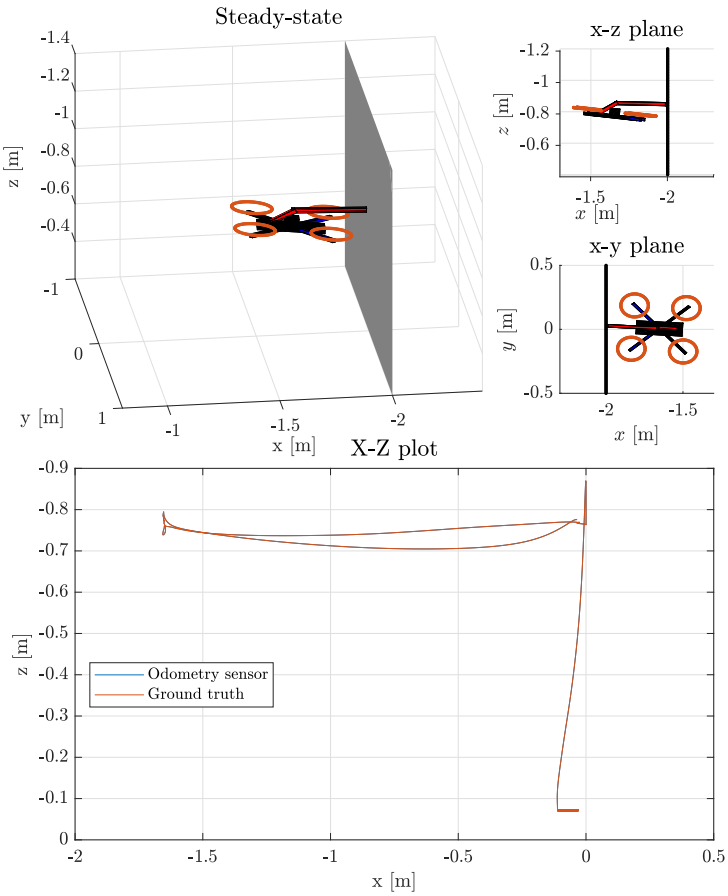


Figure C.22: Simulation with alternative configuration.
(Top) Steady-state visualization of the contact.
(Bottom) Odometry sensor data and ground truth position.

C.5 Summary

This section will summarize the main points from the simulation testing of the controller.

The noisy positions simulation shows one of the strength of this controller, as it is very robust to noise in the position in the direction of the impact. Even the scenario with a spread of 25 cm, the controller is able to maintain a force of more than 3 N for over a second. Although not included in this thesis, simulations was also carried out with bias in the position estimate, without noticeable differences from the other simulations.

In the direction parallel to the surface the controller is more sensitivity to noise during the interaction. This is natural since changing the roll angle to fix the positional error is not possible as the motion is constrained, but this action also destabilizes the interaction. The best way to solve this is to ignore the error parallel to the surface during the interaction, this requires a way of determining when the drone is interacting with the environment. One possible way to do this is using a simple force sensor; however, it might also be possible to infer this from other variables (looking at a combination of the pitch and x-acceleration).

One note is that during the previous simulations conducted in MATLAB/Simulink, a ramp function was utilized to manipulate the set-point. However, in the simulations in this part, only two set-points were used to control the interaction. One set-point is set right before the surface of interest, and when the drone has reached this point and has slowed down the set-point is changed to a point about 1 m beyond the inspection surface. There are a lot of approaches to set-point manipulation that gives successful results; however, the most important is to ensure that the horizontal forces are low when the initial contact occurs to prevent large interaction forces.

The simulations have also shown that it is difficult to stabilize the drone with a yaw angle of 0°. This is a result of the tension in the interaction, and only small disturbances will cause a rapid drift in the yaw angle. However, as soon as the yaw angle has drifted slightly, the controller compensates for it and a steady-state error is seen. Introducing integral effects to compensate for this is not a good idea, as constrained motions can cause unwanted build-up in the integrator and destabilize the interaction. A better solution is to design the hardware with a flexible joint at the end, to allow the drone to rotate slightly while the probe still maintains flat contact with the surface.

The simulations with different attack angles shows that the proposed controller is able correct these imperfect approaches. However, it should be noted that these results depend on the friction coefficient, and care should be taken when designing the hardware to ensure this. Again, it would also be beneficial to include a flexible joint near the sensor, such that the sensor gets a larger contact surface in the case of a angled approach.

The two different approaches simulated in section C.4 also show that different configurations can be utilized, without sacrificing stability.

NTNU
Norwegian University of
Science and Technology

scout
DRONE INSPECTION