Fall 7-31-2017

ROS Based High Performance Control Architecture for an Aerial Robotic Testbed

Christoph Hintz
University of New Mexico

Candidate

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

, Chairperson

ROS Based High Performance Control Architecture for an Aerial Robotic Testbed

by

Christoph Hintz

B.S., in Mechanical Engineering, Texas A&M University - Corpus Christi, 2015

THESIS

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science
Electrical Engineering

The University of New Mexico

Albuquerque, New Mexico

December, 2017

Dedication

To my parents, Grit and Mario, as well as my partner, Jasmine, for their support and encouragement they gave me along the way. Dankeschön für eure Hilfe.

"If something hasn't broken on your helicopter, it's about to." – from "Introduction to Helicopter and Tiltrotor Flight Simulation" by Mark E. Dreier

Acknowledgments

I would like to thank my grandparents, Erika and Günter Hoppe, my siblings, Marieke and Lene Hintz, as well as my closest friend Philipp Bahr, who encouraged and supported me throughout the process.

I would like to thank my academic advisor, Professor Rafael Fierro to give me the ability to do research with him, as well as supporting and advising me throughout the process. Additionally, I would like to thank Professor Francesco Sorrentino and Professor Svetlana Poroseva for their advice and being a part of my committee.

Lastly, I want to thank all the MARHES friends and members for their help during the process. I would like to name Patricio Cruz, Gregory Brunson, Steven Maurice, Shakeeb Ahmad, Corbin Wilhelmi and Joseph Kloeppel for their help and advice throughout this process.

ROS Based High Performance Control Architecture for an Aerial Robotic Testbed

by

Christoph Hintz

B.S., in Mechanical Engineering, Texas A&M University - Corpus Christi, 2015

M.S., Electrical Engineering, University of New Mexico, 2017

Abstract

The purpose of this thesis is to show the development of an aerial testbed based on the Robot Operating System (ROS). Such a testbed provides flexibility to control heterogenous vehicles, since the robots are able to simply communication with each other on the High Level (HL) control side. ROS runs on an embedded computer on-board each quadrotor. This eliminates the need of a Ground Base Station, since the complete HL control runs on-board the Unmanned Aerial Vehicle (UAV).

The architecture of the system is explained throughout the thesis with detailed explanations of the specific hardware and software used for the system. The implementation on two different quadrotor models is documented and shows that even though they have different components, they can be controlled similarly by the framework. The user is able to control every unit of the testbed with position, velocity and/or acceleration data. To show this independency, control architectures are shown and implemented. Extensive tests verify their effectiveness. The flexibility

of the proposed aerial testbed is demonstrated by implementing several applications that require high-performance control.

Additionally, a framework for a flying inverted pendulum on a quadrotor using robust hybrid control is presented. The goal is to have a universal controller which is able to swing-up and balance an off-centered pendulum that is attached to the UAV linearly and rotationally. The complete dynamic model is derived and a control strategy is presented. The performance of the controller is demonstrated using realistic simulation studies. The realization in the testbed is documented with modifications that were made to the quadrotor to attach the pendulum. First flight tests are conducted and are presented.

The possibilities of using a ROS based framework is shown at every step. It has many advantages for implementation purposes, especially in a heterogeneous robotic environment with many agents. Real-time data of the robot is provided by ROS topics and can be used at any point in the system. The control architecture has been validated and verified with different practical tests, which also allowed improving the system by tuning the specific control parameters.

Contents

List of Figures

List of Figures

List of Tables

Glossary

$\mathcal{A}$	World frame
$\mathcal{B}$	Body frame
r	Distance between world and body frame
${}^{\mathcal{A}}R_{\mathcal{B}}$	Orientation of body frame with respect to the world frame
$\mathrm{x}_{\mathcal{A}}$, $\mathrm{y}_{\mathcal{A}}$ and $\mathrm{z}_{\mathcal{A}}$	Position of quadrotor origin in world frame
ϕ, θ and ψ	Quadrotor Euler angles in world frame
$\mathrm{x}_{\mathcal{B}}$, $\mathrm{y}_{\mathcal{B}}$ and $\mathrm{z}_{\mathcal{B}}$	Position of quadrotor origin in body frame
p, q and r	Rotational angles in body frame
T_1, T_2, T_3 and T_4	Thrust of each of the four motors
L_q	Distance between quadrotor origin and motor
$\boldsymbol{\xi}$	Position vector of quadrotor in inertial frame
$\boldsymbol{\eta}$	Attitude vector of quadrotor in inertial frame
$\boldsymbol{q}$	Vector containing linear and angular position of quadrotor in inertial frame

Glossary

$\boldsymbol{v}_{\mathcal{A}}$	Linear velocity vector in inertial frame
$\boldsymbol{\nu}$	Angular velocity vector in body frame
I	Inertia matrix
I_{xx}, I_{yy} and I_{zz}	Inertia's around $\mathrm{x}_{\mathcal{B}}$, $\mathrm{y}_{\mathcal{B}}$ and $\mathrm{z}_{\mathcal{B}}$, respectively
T_f	Collective thrust from the four rotors
$\tau_{\mathrm{x}_{\mathcal{B}}}$, $\tau_{\mathrm{y}_{\mathcal{B}}}$, $\tau_{\mathrm{z}_{\mathcal{B}}}$	Moment created by motors around $\mathrm{x}_{\mathcal{B}}$, $\mathrm{y}_{\mathcal{B}}$ and $\mathrm{z}_{\mathcal{B}}$, respectively
ρ	Air density
C_t	Aerodynamic coefficient
C_d	Moment coefficient of the blade
r_b	Blade radius
ω	Rotational speed of propeller
k_t	Force constant
k_m	Moment constant
M_i	Moment created by i-th motor
M_B	Moment matrix including the moments acting on the quadrotor
u	Control matrix
m_q	Mass of the quadrotor
$\mathbf{f}$	Non-conservative forces applied to the quadrotor
$\hat{\mathbf{\Omega}}$	Skew-symmetric matrix

Glossary

g	Earth gravity constant
K_p, K_i and K_d	Tunable proportional, integral and derivative control gains, respectively
e	Error between desired and actual value
v_x	Velocity of x
T_s	Sampling time
a_{x0}, a_{x1}, a_{x2} and a_{x3}	Polynomial coefficients for x
k	Gain for trajectory tracking speed
$\mathbf{x}_L$	Position vector of pendulum
$\boldsymbol{\omega}$	Angular velocity in body frame
L_e	Distance from quadrotor origin to pendulum pivot
L_p	Pendulum length
$\{S^-\}$	Suspension point reference frame
$\{S^+\}$	Reference frame always parallel to $\{\mathcal{B}\}$
T	Kinetic Energy
V_p	Potential Energy
I_p	Pendulum Inertia
$\mathcal{L}$	Lagrangian
α	Pendulum angle
E	Energy

J	Quadratic cost function
K	Control matrix
Q	Performance index matrix
R	State cost matrix
V_1 and V_2	Lyapunov function candidates

Chapter 1

Introduction

1.1 Motivation

Unmanned aerial vehicles are becoming more popular, due to their capabilities. There is an infinite amount of possibilities where these semi-autonomous or fully-autonomous systems can assist humans to reduce time, improve quality or decrease danger. Possible applications are search and rescue, delivery, environmental monitoring, photography, etc. As they become more advanced and less expensive they also emerge into the educational sector, especially as control examples.

The quadrotor is one of the more popular unmanned aerial vehicles, due to its vertical take-off and landing capabilities, hovering flight and simple control. As a result many research groups use them as part of their testbed [2–5]. Its model has been well researched by different groups across the world [6–8], where the quadrotors are modeled in an x-configuration or plus-configuration. Attitude low level controllers have been developed to show reliable results while performing aggressive maneuvers [9, 10].

A common way to control the quadrotors is to have a ground control station, but this has the disadvantage that it lacks flexibility. The GS has to be communicating with the unmanned aerial vehicle (UAV) at all times, which requires that it be within communication range.. This makes the process of going from an indoor motion capture environment to a more realistic outdoor environment difficult. Additionally a communication delay is introduced, since the position feedback has to first go to the GS and then the GS sends control inputs to the quadrotor.

The purpose of this thesis is to develop a flexible high performance on-board control architecture for aerial vehicles in the Multi-Agent, Robotics, Hybrid, and Embedded Systems (MARHES) Laboratoy at the Univeristy of New Mexico. The goal is to have an on-board microprocessor interfaced with different vehicles in order to perform various applications, which then can be implemented reliably with minimum effort. Previosly the quadrotors in the MARHES testbed have been controlled off-board using LabView [11] or on-board using an Intel Edison microprocessor [12]. Eliminating the GS in the architecture will decrease the position feedback delay to make the UAVs more robust and high performance. A picture of the marhes testbed can be seen in Figure 1.1. The architecture makes it possible to replace the Vicon position feedback with camera position information which allows the quadrotor to fly outside the restricted area.

ROS has been evolving over the last several years as the robotic communication system of choice on the Ubuntu operating system. It has capabilities which have not been seen before. It is developed by specialists in different robotic exercises and can be used without charge by the community. It lifted robotic implementations to a whole new level. There are ROS packages for almost every robot or sensor, which can be easily manipulated and used in its node/topic environment, to implement basic to extraordinary applications. Due to its capabilities it is part of almost every robotic research laboratory and many industrial application [13–15]. Additionally it offers

the open-source physical simulator Gazebo, which offers detailed models of almost every robot. The controllers can be implemented and tested in Gazebo and copied almost one to one to the real system, which makes the whole process faster and safer. The smaller, lighter and faster microprocessors, like Odroid, Rasperry Pi or Nvidia Jetson allow for the on-board use of Ubuntu and ROS resources. This makes the robot more interesting due to its advanced capabilities of on-board processing and elimination of a GS, if desired.

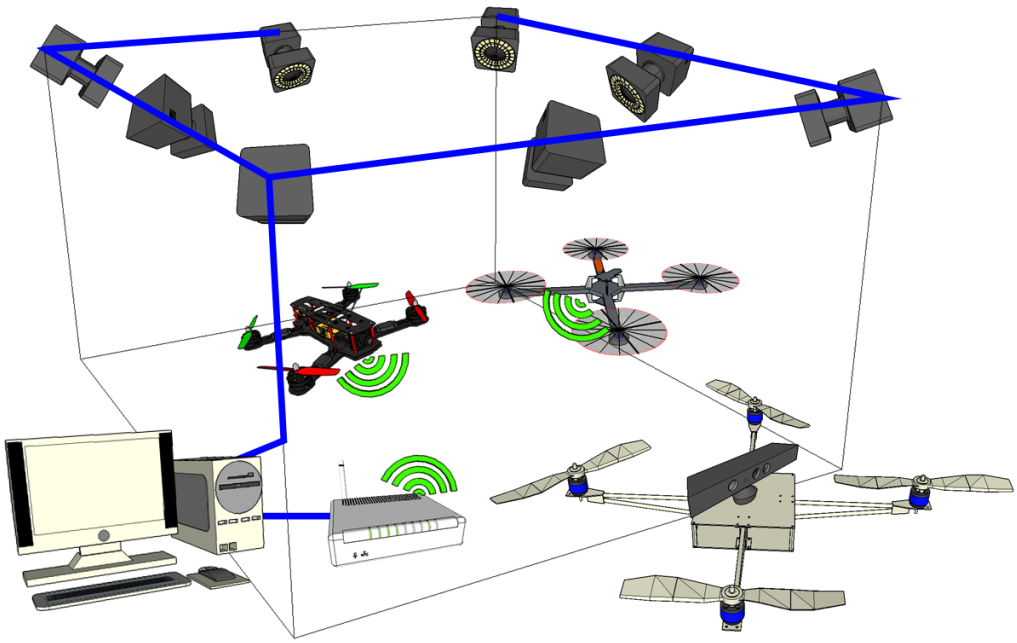


Figure 1.1: Drawing of the MARHES testbed with quadrotors in the Vicon motion capture area. The UAV in the front is able to go out of the Vicon area by getting position information from a camera.

1.2 Problem Statement

The main goal of this thesis is to implement a real-time high performance control architecture for UAV's. To achieve this, a ROS based high level architecture has to

be implemented for an on-board microprocessor. This allows for independence of the vehicle type. It is possible to use an AscTec AutoPilot or Pixracer flight controller with any kind of configuration of the body, as long as the attitude controller is accordingly tuned and the high level controllers are similar. The interface setup allows for extraction and use of real-time data from the quadrotor or motion capture system at any point. The quadrotor must be able to be controlled with position, velocity and/or acceleration commands.

To achieve this goal, certain subgoals have to be met along the process. A model of a rigid body quadrotor is derived. Linear position control has been developed and implemented. The control has to demonstrate that it is able to takeoff, land, hover, follow waypoints, follow velocities and track trajectories. The ROS framework has to be able to interface with MATLAB and execute trajectories that were developed in Matlab. It also has to fly in a stable manner, even with off-centered balance. It will be evaluated with different applications that were developed to demonstrate the flexibility of the vehicle. A framework for a flying inverted pendulum will be developed and shown in Gazebo, to explore the possibility of implementing it in the MARHES laboratory.

1.3 Related Work

In the past few years, the quadrotor has become more popular and has been well studied in papers that show many types of stabilization techniques, controllers and models. The nonlinear model, which assumes a rigid body configuration has been derived using Newton-Euler formalism or Lagrangian [6]. This work describes in detail the dynamic model of a x-configuration quadrotor with all forces, moments, inertias, etc., that act on it. The difference between a x-configuration and plus-configuration is in the control matrix [7]. Usually the low level controller is stabilizing

the attitude. It can be seen that there is a tendency towards PD, PID controllers or nonlinear controllers. All three variations have proven to be good solutions and can be used in aggressive maneuvers [16, 17].

Different applications of quadrotors have been exploited. As an example, loads have been modeled hanging from the aerial vehicle to reduce swinging [18]. Agile adaptive control is implemented to correct the switch in the center of mass. These systems can be used in aerial transportation. It has been simulated and tested in the MARHES Testbed. Additionally quadrotors could be used for carrying information between ground agents. Many micro robots could be implemented in order to take pictures or collect information about the environment. The quadrotors could be used in this environment to collect the data from the ground-robot via an optical link and relay it to a home station [19]. A lot of research has been devoted to reachability analysis, where the quadrotor or a group of quadrotors could catch a potential threat [20, 21].

Additionally, a quadrotor could be used as an aerial Segway or flying taxi. A model of an inverted pendulum on a quadrotor would be needed for this type of application, which has been done for a linear and rotational inverted pendulum. Swing-up and balancing controllers have been developed for both cases, where it is usual to swing-up the pendulum with an energy controller and balance it with an LQR controller [22–25]. Researchers at ETH Zurich have already implemented flying inverted pendulums, which are balanced in the center of the quadrotor. The UAV's are able to balance the pendulum as well as throw and catch it from one quadrotor to another, respectively [26, 27]. For the throwing maneuver an optimal path of the quadrotor to the pendulum launch is derived, as well as an optimal catching instant. The simulation and experiments have been published and demonstrated in the Flying Machine Arena.

All the work that is presented in this sections show the ability and flexibility of

quadrotors. A desire to have a general architecture which can be used for various applications to simplify the implementation process as well as the performance can be seen.

1.4 Organization of this Thesis

In the following chapter the different hardware and software used throughout the thesis is presented. It includes detailed descriptions of vehicles, their sensors, the motion capture system, on-board microprocessor and the Robotic Operating System. Their characteristics and advantages/disadvantages will be shown. In Chapter 3 the model of the quadrotor will be derived using Newton-Euler formalism. The differences between a x-configuration and plus-configuration will be explained. Additionally, it will describe the ROS architecture and implementation of the on-board microprocessor. Chapter 4 shows different applications that demonstrate the flexibility of this architecture. Chapter 5 will show the development of a control strategy that will be able to swing up and balance a flying inverted pendulum in a linear and rotational manner. It includes the model, controllers for swing-up and balancing, stability analysis using Lyapunov, Gazebo implementation and preliminary implementation results. Chapter 6 will be a conclusion and discussion of future possibilities of the framework.

Chapter 2

System Overview

2.1 Hardware

2.1.1 MARHES Testbed

The heterogeneous MAHRES Testbed at the University of New Mexico consists of a variety of different ground and aerial vehicles. The ground rovers are three TurtleBot2 [28], five Pioneer 3-AT [29] and ten TXT-1 body based monster trucks [30]. Additionally there are currently two different bio-inspired crawling robots, the OctoRoACH [31] and five miniROaCHes [32], which are a modification of the Kamigami Dash robot. The flying robots used in the MARHES testbed are three AscTec Hummingbirds [33], seven Pixracer based quadrotors with QAV250 frame and a custom made Pixracer based multirotor. Furthermore, two humanoid Baxter [34] robots are part of the laboratory. For exact position feedback a Vicon MX [35] motion capture system is used. A safety net is installed in the testbed area for security reasons. The MARHES laboratory, with a majority of the robots, can be seen in Figure 2.1. Odroid XU-4 [36] or Nvidia Jetson TK1 [37] are used as on-board

microprocessors for the aerial vehicles, whereas the miniRoACHes are developed with a Raspberry Pi Zero W [38] as the on-board computer.

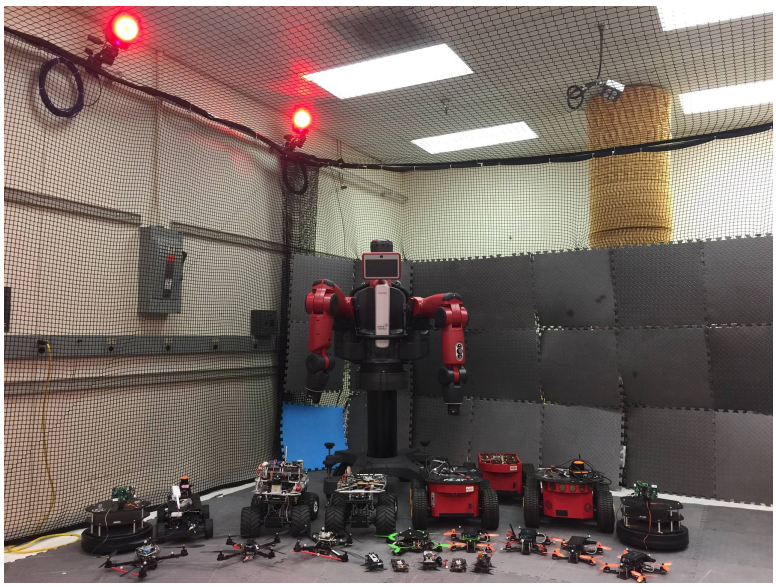


Figure 2.1: Heterogenous MARHES Testbed with a variety of humanoid, ground and aerial robots in the Vicon motion capture area.

2.1.2 AscTec Hummingbird

The AscTec Hummingbird quadrotor is typical vehicle used in research due to its advanced capabilities, open-source software and extensive resources that are provided by research groups on-line. The UAV can be seen in Figure 2.2. Designed for research, it has the capability of high performance flight maneuvers and, due to its design, is easy to repair [33]. It is designed to fly in a plus configuration, which means that the front extends along one of the arms, which can be seen by the orange tape attached in the picture.

It comes with the Ascending technologies AutoPilot, which is able to run Low-Level(LL) attitude control loops including the process of collected sensor data with

data fusion at 1000Hz. It is possible to run custom High-Level controllers on the provided High-Level(HL) processor, however for this thesis the HL processor is only used to interface the Odroid-XU4 microprocessor and pass the desired inputs to the LL controller. The board can be interfaced with GPS provided by AscTec for outdoor implementations. For safety, AscTec is providing several emergency modes that can be essential to rescue the vehicle before a disaster happens. The HL processor provides a serial interface which is used to get information from radio receivers, XBee receivers or other serial interfaces. For this thesis it will communicate with the Odroid to implement position, velocity or acceleration control at a bau drate of 921600.

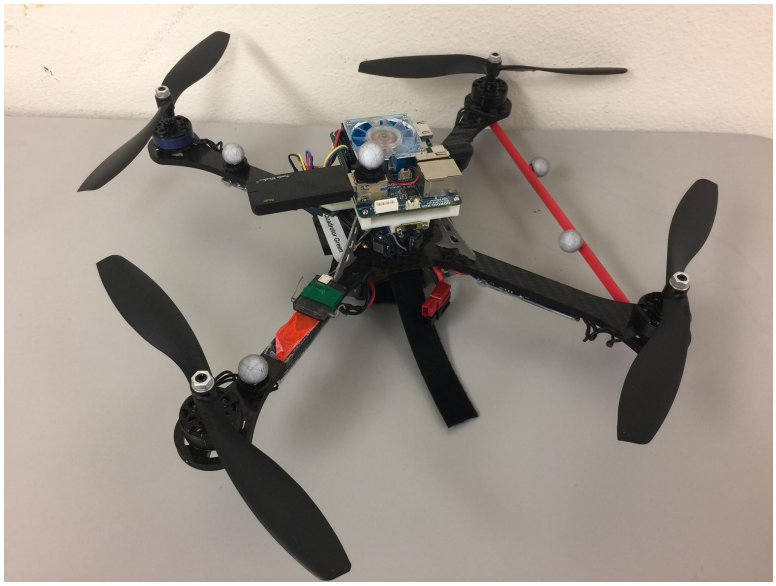


Figure 2.2: AscTec Hummingbird quadrotor with Odroid XU4 microprocessor on top of it and Vicon markers attached.

The Hummingbird uses specific Electronic Speed Controllers(ESC) and motors developed for it. This guarantees flawless interfacing with the AutoPilot. The motors are designed for improving efficiency as well as maneuverability in combination with the ESC's. The 1000kV motors paired with the 12 inch propellers only need 100W

per motor. This implementation makes it possible to reach flight times of up to 30 minutes if there is no payload attached to the quadrotor. Each motor is able to approximately generate 5N of maximum thrust. The maximum takeoff weight for the quadcopter is given to be 710g, and having a weight of 510g it allows for 200g of extra payload that can be attached safely. The batteries used for the UAV are 3s Lipo batteries, which are light but also less powerful than higher cell batteries. The 3 Cell batteries provide a voltage of 12.6V when they are fully charged. The batteries used for this thesis have a rating of 1800mAh.

Additionally, the body structure is very lightweight, which also helps to increase efficiency as well as a agility. The core of the hummingbird is made out of three levels of carbon fiber plates for stiffness and weight reduction. These levels are connected by four magnesium structures which add stability to the frame. The arms are made out of two carbon fiber plates which are connected through a layer of balsa wood. This makes them structurally stiff, but the main advantage is that the wood layer makes them extremely light weight. The diameter from motor to motor is 36.5mm which makes it a Mini Rotorcraft Unmanned Aerial or Aircraft System(RUAS) [39].

2.1.3 QAV250

The QAV250 based quadrotor has been developed as part of a project with Sandia Laboratories. It was developed to have a small UAV with extensive maneuverability and payload capacities. The vehicle has been built from scratch, which allowed for a variety of components to choose from to produce the best output. It can be seen in Figure 2.3. It flies in a x-configuration.

For the flight controller, the PixRacer is used due to its compatibility with ROS and its extensive resources [40]. It is able to communicate with the on-board computer over USB. Its attitude controller is running at 400Hz, whereas it is getting

Accelerometer, Gyroscopic and Magnetometer updates at 4000Hz. One of the main advantages is that it is compatible with OneShot ESC protocol. This increases the flight performance, because the Flight Controller(FC) and ESC synchronize their loop-time, which eliminates the error of reused PWM signals without OneShot. Furthermore, the signal sent from FC to ESC is shorter, 125us-250us compared to 1ms-2ms. With OneShot protocol comes a feature called Active Braking, this will actively brake the motor when the speed is supposed to be decreased by applying reversed voltage, instead of applying less volts and waiting until the desired propeller speed is reached. These factors lead to a increase in quadrotor response, which adds stability, agility and maneuverability. Additionally, it has several telemetry, GPS and radio receivers that work with almost all equipment that is used in the Radio Controlled(RC) vehicle sector. Its small footprint (32mmx32mm) allows it to be implemented in small body frames.

Figure 2.3: QAV250 frame based quadrotor with Odroid XU4 microprocessor on top of it and Vicon markers attached. The Odroid is secured by a custom design and 3D printed part.

The ESC used to convert the FC signals to motor outputs are Lumenier F390

30A, which come with OneShot and Active Breaking capabilities, when configured on the FC side. They withstand 30A continuous and 40A burst for 10 seconds. Its powerful F390 chip allows for fast response. Cobra 2205 size motors with 2500kV were choosen due to their quality and extreme thrust generation. Their advanced bearings allow almost frictionless rotation. In combination with 5 inch propellers and 16V they are able to produce 1.5kg thrust per motor. The combined maximum thrust of 6kg is an advantage, since the complete quadrotor weights approximately 820g with battery it is able to handle securely an additionally 3kg payload.

The Lumenier QAV250 frame is a two level x-type frame. It has a motor to motor diameter of 250mm. The 3k carbon fiber structure adds stiffness to the quadrotor in addition to being light-weight. The design allows the quadcopter to withstand even strong impacts. Considering the size, weight and payload capability it can be counted as a small-scale RUAS [39].

2.1.4 Vicon System

The MARHES laboratoy Vicon system is used for position and orientation feedback for the aerial vehicles. The high precision, fast update rate, simple setup, friendly user interface and extensive resources make it a perfect research resource to implement, test and tune control algorithms. It consist of 8 cameras and an MX Giganet which collects all of the camera feedback and interfaces a computer running Vicon Tracker software on Windows 7 Operating System(OS) [35]. It delivers high-precision vehicle data with an error down to 0.5mm in the position and 0.5deg to the rotation at an update rate of up to 225Hz. The Bonita cameras operate at 240 frames per second(fps). The cameras have red LED's which are reflected by the markers, as seen in Figure 2.2 and 2.3, to get feedback of a predefined object. The setup schematic of the Vicon area can be seen in Figure 2.4. In the picture the Ethernet connections

are shown by the solid lines and the Wifi connection can be seen by the dotted line.

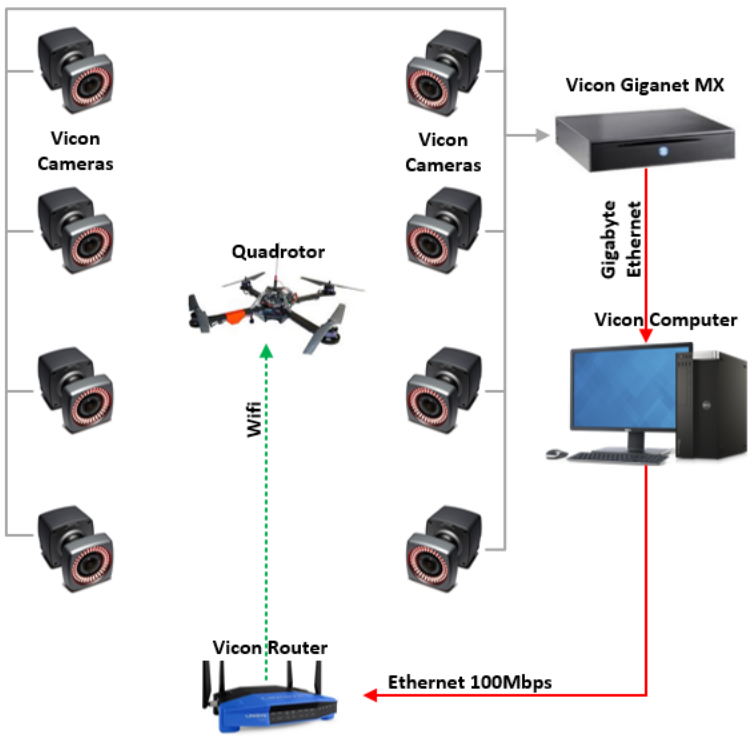
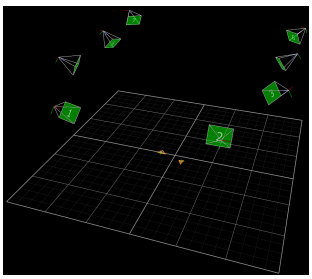


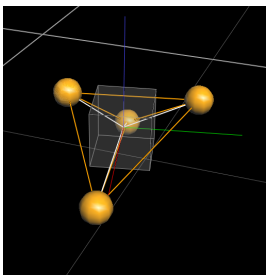
Figure 2.4: Schematic of the Vicon system and its setup in the laboratory.

The Tracker software is the interface between the computer and the MX Giganet. First the cameras have to be calibrated using the Vicon Wand seen in Figure 2.5a. Next the user is able to set the origin with the same wand, where the long end marks the x-axis. The coordinate system follows a East-North-Up system, which means that z is up, y to the left and x is to the front. The markers have to be attached on the robots next in an asymmetric order, so that Vicon is able to see differences between the agents. It is then possible to create a model in the Tracker software, which can be published to the network. Every robot that is connected to the same router via Ethernet or wireless is able to run ViconStream code, which is provided for Windows, Ubuntu and MAC OS by Vicon. A picture of the virtual created room in Tracker can be seen in Figure 2.5b and a close range view of the created Vicon

(a) Vicon Wand to setup the Vicon area.



(b) Vicon area showing the created area, each of the cameras and 2 objects.



(c) Close look at an object in the tracker software.

Figure 2.5: Vicon equipment for setup and view of Tracker software.

model in the real-time position and orientation tracking system is seen in Figure 2.5c. It shows a close range view of the object created with the markers and the origin for this object. The position and orientation of this origin is the published by Vicon, and it is possible to manipulate the origin of an object if needed.

2.1.5 Odroid XU4

The Odroid XU-4, which can be seen in Figure 2.6, has been chosen due to its enhanced capabilities and small size [36]. Some relevant characteristics given by the developer are

- Cortex™-A15 2Ghz and Cortex™-A7 Octa core CPUs
- 2Gbyte LPDDR3 RAM PoP stacked
- eMMC5.0 HS400 Flash Storage
- 2 x USB 3.0 Host, 1 x USB 2.0 Host
- Size : 83 x 58 x 20 mm

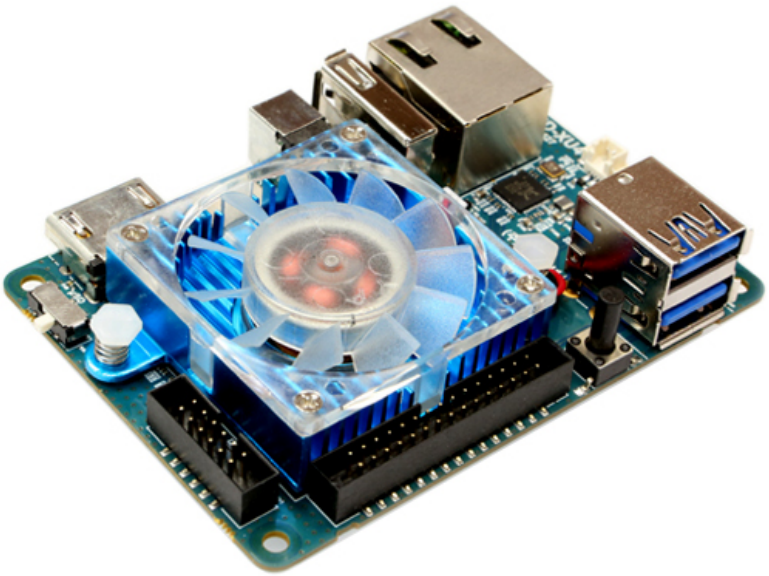


Figure 2.6: Odroid XU4 microprocessor from Hardkernel.

- Linux Kernel 4.9 LTS

The 64 bit memory bandwidth processor in combination with the 2Gb RAM gives enough calculation power to run advanced programs, especially for an on-board computer. Similar microprocessors are significantly less powerful, as an example the Odroid is approximately 7 times faster than the latest Raspberry PI 3. For this thesis 32gb eMMC cards are used, which are approximately 4 times faster than high quality MicroSD cards. The storage makes it possible to use Lubuntu 14.04 as the operating system with a desktop version of ROS. The serial connector gives the ability to connect the Odroid XU4 with the AscTec AutoPilot board and the USB 3.0 port is used to connect to the Pixracer FC. In both cases, one of the USB 3.0 port is used for the WiFi adapter to get fast high precision Vicon position and orientation updates from the Router. The small footprint of the microprocessor makes it possible to fit it on small quadrotor bodies. Parts to mount the Odroid on the different multirotor bodies have been designed and printed. The light weight of 38 grams does not add a lot of payload to the UAV's, which makes it a reasonable choice.

2.2 Software

2.2.1 ROS

The Robotic Operating System has become the standard for robotic research. It is an open source free of charge meta-operating system [41]. It combines the following services

- operating system
- hardware abstraction
- low-level device control
- message-passing between processes
- package management

The robot framework additionally provides libraries to obtain, build, write and run code. The communication infrastructure is the main advantage of ROS. It makes it as simple as possible to use different robots or sensors and interface them with each other. It is a node and topic system. Nodes are essentially `Python Code` or `C++` programs downloaded or specifically built by the users which can subscribe or publish to different topics. As soon as a topic is published each node has the ability to subscribe to it. This concept can be seen in Figure 2.7. The ROS Message Types are predefined or custom messages, which define the format of how the information is published. There are basic messages which are efficient and well arranged, however advanced users can also create their own message types if needed.

ROS was designed to interface with Ubuntu, which is used as a typical research operating system, due to its capabilities and the fact that it is free of charge. Over the

Figure 2.7: Publish/Subscribe concept of topics by nodes in the Robotic Operating System [1]

years there have been several distribution versions of ROS. A table of ROS version, their Ubuntu compatibility and end of support date can be seen in Table 2.1. The ROS version used throughout the thesis is ROS Indigo Igloo. It has the advantage that it is mainly made for Ubuntu 14.04, which is a Long Term Support version of Ubuntu that added security to the project. Ubuntu 14.04 also has a light version, called Lubuntu which interface with the Odroid-XU4. The user has the ability to install different versions of ROS, which are the Full Desktop, Desktop, Bare-Bones and Individual package version. The difference between them are essentially the amount of packages that come predefined with them, which correlates to the storage amount needed for the version. Whereas on a ground station one would install the Full-Desktop version, for the on-board computer, the desktop version has been chosen. It ensures that not to much of the 32gb memory storage of the Odroid XU4 eMMC card is used and is still sufficient. Additionally, specific packages are needed later on they can be installed manually, so there is no real drawback of using a lighter version other than the compiling time for ROS packages may take longer, due to initially missing packages.

For the communication between the Odroid XU4 and AscTec AutoPilot the `asctec_mav_framework` ROS package is been used. This package is supported up

ROS Version	Ubuntu Version	End of Support
Lunar Loggerhead	17.04(Zesty),16.10(Yakkety)& 16.04(Xenial)	May 2019
Kinetic Kame	16.04(Xenial) & 15.10(Wily)	April 2021
Jade Turtle	15.04(Wily), 14.10(Utopic) & 14.04(Trusty)	May 2017
Indigo Igloo	14.04(Trusty) & 13.10(Saucy)	April 2019
Hydro Medusa	13.04(Raring), 12.10(Quantal) & 12.04(Precise)	May 2015
Groovy Galapagos	12.10(Quantal), 12.04(Precise) & 11.10(Oneiric)	July 2014

Table 2.1: ROS Distributions since December 31, 2012. Additionally, information about the compatibility with Ubuntu versions and end of date for their support.

to ROS Indigo. It provides communication at baud rates of 921600. Additionally, IMU data fusion at 1kHz has been implemented in the package and is able to be used for the attitude control. These state estimations are provided in custom ROS messages which can be used to implement controllers on the platform. The framework allows to send positions, velocities or accelerations to the quadrotor. Positions and velocities are in the world frame which is provided by the Vicon system. For the acceleration, the user sends a normalized thrust value (0 to 1), roll and pitch in body frame in rad and the heading in $\frac{rad}{s}$. It connects via serial communication. Since the Odroid has a 1.8V logic and the AutoPilot board has a 3.3V logic, a simple voltage level converter board is used to make communication possible. For The PixRacer FC, the `mavros` package is used. It is developed for `px4` based flight controllers, like PixHawk, PixHawk Mini and PixRacer. The advantages of this package are the extensive resources and number of people involved in the project. It is possible to send motion capture or vision position data to the ROS package and fuse it with the IMU data with the desired trust in the on-board and off-board data. The quadrotor can be controlled in various ways by sending linear and rotational positions, velocities and/or accelerations to the PixRacer. An Extended Kalman Filter (EKF) provides state estimation. The motion capture system is using the Vicon SDK in combination with a lightweight communications and marshaling package to optimize the high-bandwidth communication.

Additionally ROS provides a physical simulator which is called Gazebo [42]. Features of Gazebo are

- Dynamics simulation - high performance physics engines
- Advanced 3D graphics - realistic rendering of environment
- Sensors and noise - model sensors with noise
- Plugins - ability to develop custom plugins for robot, sensor and environmental control
- Robot models - many existing robots have already been modeled and are available
- TCP/IP transport - ability to run simulation from remote servers

Since Gazebo is embedded in ROS, it uses the same structure. This makes it a useful tool, because the controllers can be created, implemented and tested in a close to real world scenario and taken to the real robot in similar fashion. These advantages lead to the use of the simulator for a specific application presented in this thesis. The simulation will consider an inverted pendulum on a quadrotor and has been completely tested in Gazebo. The `rotors_simulator` package has been used to get an exact replica of the AscTec Hummingbird with all its characteristics [43]. This eliminated the need of designing the quadrotor from zero in Gazebo. A pendulum has been modeled and attached to the quadrotor with all characteristics and inertias. The cascade control strategy has been implemented and is explained later on in the thesis.

Additionally a new ROS version is currently under development. Even though new distributions of ROS are continuously developed, there were some disadvantages that are so deep in the ROS core that a new version called ROS 2 is being developed.

This new version is focusing on improving support for multiple Data Distribution Service(DDS) middleware implementations. It is being tested for more Operating Systems, including Windows 10. It also tries to implement more programming languages. All these changes are an effort to improve the system and further adapt to the robotics community, which has changed since ROS 1 was first introduced in 2007.

Chapter 3

Modeling of the Quadrotor

For the quadrotor model certain assumptions have to be made. The first assumption is that the quadrotor is symmetrical and has a rigid body structure with 6 Degrees of Freedom (DOF) [6, 16, 44, 45]. 3 DOF for translation and 3 for orientation. Additionally, the propellers are also assumed to be a rigid body. The thrust and drag created by the propellers are proportional to the square of the rotational propeller speed. The center of mass (COM) of the quadrotor lies at the center of its body frame.

3.1 Kinematics Model

The schematics of the model can be seen in Figure 3.1. The world frame $\{\mathcal{A}\}$ has the reference axes $x_{\mathcal{A}}$, $y_{\mathcal{A}}$ and $z_{\mathcal{A}}$. It can be noticed that $z_{\mathcal{A}}$ is going out of the world or up. The Euler angles which define the rotation around the axes are defined as follows, ϕ is the angle around the x-axis, θ is the angle around the y-axis and ψ is the angle around the z-axis. The body frame $\{\mathcal{B}\}$ of the the quadrotor is attached to its COM and has the reference axes $x_{\mathcal{B}}$, $y_{\mathcal{B}}$ and $z_{\mathcal{B}}$. The axes $x_{\mathcal{B}}$ is pointing toward propeller

one and represents the front of the quadrotor, $y_{\mathcal{B}}$ is pointing toward propeller two and $z_{\mathcal{B}}$ i pointing upwards. The distance between these two coordinate frames is r, where r can be expressed as $r = [x\ y\ z]^T$.

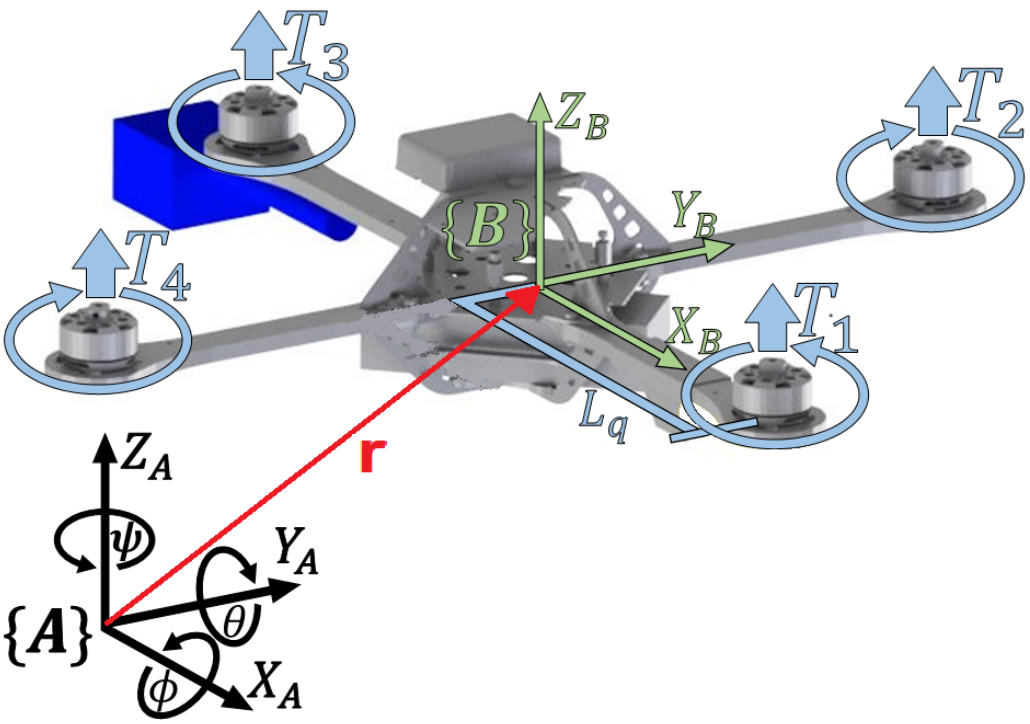


Figure 3.1: Schematic diagram of the quadrotor aerial vehicle. The world frame $\{\mathcal{A}\}$ shown in black and the body frame $\{\mathcal{B}\}$ shown in green with their related axes. The distance from the origin of the world frame to the origin of the body frame is shown by r in red. The system characteristics are symbolized in light blue.

The multirotor inputs for the four motors are represented as T_1, T_2, T_3 and T_4. T_1 and T_3 are rotating counter-clockwise, whereas T_2 and T_4 are rotating clockwise. The distance between the center of mass of the quadrotor and the motors is shown by L_q.

The rotational matrix ${}^{\mathcal{A}}R_{\mathcal{B}}$ defines the orientation of the body frame to the world frame, which is needed due to the fact that some states of the dynamic model are

measured in the world frame (i.e. gravitation and quadrotor position), and some are measured in body frame (i.e. thrust created by propellers) [44]. The rotation sequence used for this thesis will be Z-Y-X or 3-2-1. The quadrotor orientation is described by roll (ϕ), pitch(θ) and yaw(ψ), so that the rotational matrix ${}^{\mathcal{A}}R_{\mathcal{B}}(\psi, \theta, \phi)$ looks as follows

$$ {}^{\mathcal{A}}R_{\mathcal{B}} = \begin{bmatrix} \mathrm{c}\theta\mathrm{c}\psi & \mathrm{s}\theta\mathrm{s}\phi\mathrm{c}\psi - \mathrm{c}\phi\mathrm{s}\psi & \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\psi \\ \mathrm{c}\theta\mathrm{s}\psi & \mathrm{s}\theta\mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{c}\psi & \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{s}\phi\mathrm{c}\psi \\ -\mathrm{s}\theta & \mathrm{s}\phi\mathrm{c}\theta & \mathrm{c}\phi\mathrm{c}\theta \end{bmatrix}. \tag{3.1} $$

It can be noted that s is the sine function and c is the cosine function. Additionally it can be said that ${}^{\mathcal{A}}R_{\mathcal{B}}$ is orthogonal.

3.2 Dynamic Model of a Quadrotor

The position of the quadrotor COM is defined in the inertial frame by $\boldsymbol{\xi}$ and the attitude is expressed by $\boldsymbol{\eta}$, so that the vector $\mathbf{q}$ accommodates the linear and angular position in the inertial frame

$$ \boldsymbol{\xi} = \begin{bmatrix} \mathrm{x}_{\mathcal{A}} \\ \mathrm{y}_{\mathcal{A}} \\ \mathrm{z}_{\mathcal{A}} \end{bmatrix}, \boldsymbol{\eta} = \begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix}, \mathbf{q} = \begin{bmatrix} \boldsymbol{\xi} \\ \boldsymbol{\eta} \end{bmatrix}. \tag{3.2} $$

The body dynamics of the quadrotor are represented by $\mathbf{v}_{\mathcal{A}}$, which represents the linear velocity in inertial frame and $\boldsymbol{\nu}$, which represents the angular velocity in body frame.

$$ \mathbf{v}_{\mathcal{A}} = \begin{bmatrix} \dot{\mathrm{x}}_{\mathcal{A}} \\ \dot{\mathrm{y}}_{\mathcal{A}} \\ \dot{\mathrm{z}}_{\mathcal{A}} \end{bmatrix}, \boldsymbol{\nu} = \begin{bmatrix} \mathrm{p} \\ \mathrm{q} \\ \mathrm{r} \end{bmatrix}, \tag{3.3} $$

where p, q and r represent the angular velocities in the fixed body frame. The inertia matrix for the quadrotor is defined as follows

$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}. \tag{3.4}$$

The inputs to the system are the four forces created by the motors [46]. The first input is the collective thrust represent by T_f.

$$T_f = \sum_{i=1}^{4} T_i \tag{3.5}$$

The force created by a single rotor can be described as follows.

$$T_i = \frac{1}{2}\rho A C_t r_b^2 \omega_i^2 = k_t \omega_i^2 \tag{3.6}$$

where ρ represents the air density, C_t is the aerodynamic coefficient, A is the blade area, r_b is the blade radius, k_t represents the force constant and ω is the rotational speed of the blade. With the help of equation (3.6) it is possible to rewrite equation (3.5) to

$$T_f = k_t \sum_{i=1}^{4} \omega_i^2 \tag{3.7}$$

The second and third input into the system represent the torques around the x and y body axis created by the difference of motor speeds. The moment, which is force multiplied with the distance, around the body axis $\mathrm{x}_{\mathcal{B}}$ and $\mathrm{y}_{\mathcal{B}}$ can be described as follows

$$\begin{aligned}\tau_{\mathrm{x}_\mathcal{B}} &= T_2 L_q - T_4 L_q \\ &= k_t \omega_2^2 L_q - k_t \omega_4^2 L_q \\ &= k_t L_q (\omega_2^2 - \omega_4^2)\end{aligned} \tag{3.8}$$

$$\begin{aligned}\tau_{\mathrm{y}_\mathcal{B}} &= T_3 L_q - T_1 L_q \\ &= k_t \omega_3^2 L_q - k_t \omega_1^2 L_q \\ &= k_t L_q (\omega_3^2 - \omega_1^2).\end{aligned} \tag{3.9}$$

The fourth input describes the moment around the z body frame axis. It is is calculated by addition/subtraction of the individual moments created by each rotor. The moments created by each motor can be described as follows

$$M_i = \frac{1}{2} \rho A C_d r_b^2 \omega_i^2 = k_m \omega_i^2. \tag{3.10}$$

The moment created by the i-th motor is described by M_i, C_d represents the moment coefficient of the blade and k_m describes the moment constant. This equation helps to represent the moment around $\mathrm{z}_\mathcal{B}$ which looks like

$$\begin{aligned}\tau_{\mathrm{z}_\mathcal{B}} &= M_2 + M_4 - M_1 - M_3 \\ &= k_m \omega_2^2 + k_m \omega_4^2 - k_m \omega_1^2 - k_m \omega_3^2 \\ &= k_m (\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2).\end{aligned} \tag{3.11}$$

It is known that u_4 is the weakest input, due to the fact that the moment constant k_m is usually smaller than the force constant k_t. Combining equation (3.5), (3.8), (3.9) and (3.11) one can develop a matrix, which includes the forces and moments acting on the quadrotor by its inputs, which looks as follows

$$M_B = \begin{bmatrix} T_f \\ \tau_{\mathrm{x}_\mathcal{B}} \\ \tau_{\mathrm{y}_\mathcal{B}} \\ \tau_{\mathrm{z}_\mathcal{B}} \end{bmatrix} = \begin{bmatrix} k_t \sum_{i=1}^{4} \omega_i^2 \\ k_t L_q(\omega_2^2 - \omega_4^2) \\ k_t L_q(\omega_3^2 - \omega_1^2) \\ k_m(\omega_2^2 + \omega_4^2 - \omega_1^2 - \omega_3^2) \end{bmatrix}. \tag{3.12}$$

This can be redefined into the inputs for the system, where T_f is u_1, $\tau_{\mathrm{x}_\mathcal{B}}$ represent u_2, $\tau_{\mathrm{y}_\mathcal{B}}$ defines u_3 and $\tau_{\mathrm{z}_\mathcal{B}}$ is u_4. This allows to express the control matrix u as follows.

$$u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} k_t & k_t & k_t & k_t \\ 0 & k_t L_q & 0 & -k_t L_q \\ -k_t L_q & 0 & k_t L_q & 0 \\ -k_m & k_m & -k_m & k_m \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}. \tag{3.13}$$

3.2.1 Newton-Euler Approach

Using Newton-Euler formalism, the equations of motion for the translational and rotational motion can be derived. The translational change is described in world frame, whereas the rotational motion is described in body frame [45]. The external forces applied to a rigid body can be described by the following equations

$$\dot{\boldsymbol{\xi}} = \mathbf{v}_\mathcal{A} \tag{3.14}$$

$$m_q \dot{\mathbf{v}}_\mathcal{A} = \mathbf{f} \tag{3.15}$$

$$\dot{R} = R\hat{\boldsymbol{\Omega}} \tag{3.16}$$

$$\boldsymbol{I}\dot{\boldsymbol{\nu}} = -\boldsymbol{\nu} \times \boldsymbol{I}\boldsymbol{\nu} + \tau. \tag{3.17}$$

Equation (3.15) describes Newton's first law applied to the body frame, where $\mathbf{f}$ described the non-conservative forces applied to the quadrotor and m_q denoted the

quadrotor mass. The following equation describes the law to achieve the derivative to the rotational matrix, with respect to time, where $\hat{\mathbf{\Omega}}$ represents the skew-symmetric matrix. Equation (3.17) is used to calculate the moments around the body axis. The non conservative forces can be separated in gravitational and translational forces, so that $\mathbf{f} = {}^{\mathcal{A}}R_{\mathcal{B}}T_f - m_q g\boldsymbol{\xi}_z$, and equation (3.15) becomes

$$m_q\dot{\mathbf{v}}_{\mathcal{A}} = {}^{\mathcal{A}}R_{\mathcal{B}}T_f - m_q g\boldsymbol{\xi}_z. \tag{3.18}$$

The translational forces of equation (3.18) were calculated using the statement ${}^{\mathcal{A}}R_{\mathcal{B}}T_f$, where $T_f = [0 \quad 0 \quad k_t\sum_{i=1}^{4}\omega_i^2]' = [0 \quad 0 \quad u_1]'$ according to the input matrix represented in Equation (3.13), so that they can be represented as follows

$$\begin{aligned} {}^{\mathcal{A}}R_{\mathcal{B}}T_f &= \begin{bmatrix} \mathrm{c}\theta\mathrm{c}\psi & \mathrm{s}\theta\mathrm{s}\phi\mathrm{c}\psi - \mathrm{c}\phi\mathrm{s}\psi & \mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\psi \\ \mathrm{c}\theta\mathrm{s}\psi & \mathrm{s}\theta\mathrm{s}\phi\mathrm{s}\psi + \mathrm{c}\phi\mathrm{c}\psi & \mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{s}\phi\mathrm{c}\psi \\ -\mathrm{s}\theta & \mathrm{s}\phi\mathrm{c}\theta & \mathrm{c}\phi\mathrm{c}\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ u_1 \end{bmatrix} \\ &= \begin{bmatrix} (\mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\psi)u_1 \\ (\mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{s}\phi\mathrm{c}\psi)u_1 \\ (\mathrm{c}\phi\mathrm{c}\theta)u_1 \end{bmatrix}. \end{aligned} \tag{3.19}$$

The gravitational forces only act on the altitude or z component, so that

$$\begin{aligned} m_q g\boldsymbol{\xi}_z &= m_q g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \\ m_q g \end{bmatrix}. \end{aligned} \tag{3.20}$$

With equation (3.19) and (3.20) the non conservative forces for the translational motion can be rewritten as

$$\ddot{\mathrm{x}}_{\mathcal{A}} = \frac{(\mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\psi)u_1}{m_q}, \tag{3.21}$$

$$\ddot{\mathrm{y}}_{\mathcal{A}} = \frac{(\mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{s}\phi\mathrm{c}\psi)u_1}{m_q}, \tag{3.22}$$

$$\ddot{\mathrm{z}}_{\mathcal{A}} = \frac{(\mathrm{c}\phi\mathrm{c}\theta)u_1}{m_q} - g. \tag{3.23}$$

For the rotational motion we can rewrite equation (3.17) to

$$\begin{aligned}
\begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \dot{\mathrm{p}} \\ \dot{\mathrm{q}} \\ \dot{\mathrm{r}} \end{bmatrix} &= - \begin{bmatrix} \mathrm{p} \\ \mathrm{q} \\ \mathrm{r} \end{bmatrix} \times \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \mathrm{p} \\ \mathrm{q} \\ \mathrm{r} \end{bmatrix} + \begin{bmatrix} \tau_{\mathrm{x}_{\mathcal{B}}} \\ \tau_{\mathrm{y}_{\mathcal{B}}} \\ \tau_{\mathrm{z}_{\mathcal{B}}} \end{bmatrix} \\
&= \begin{bmatrix} \mathrm{qr}I_{yy} - \mathrm{rq}I_{zz} \\ \mathrm{rp}I_{zz} - \mathrm{pr}I_{xx} \\ \mathrm{pq}I_{xx} - \mathrm{qp}I_{yy} \end{bmatrix} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} \\
&= \begin{bmatrix} \frac{\mathrm{qr}(I_{yy} - I_{zz}) + u_2}{I_{xx}} \\ \frac{\mathrm{rp}(I_{zz} - I_{xx}) + u_3}{I_{yy}} \\ \frac{\mathrm{pq}(I_{xx} - I_{yy}) + u_4}{I_{zz}} \end{bmatrix}.
\end{aligned} \tag{3.24}$$

For small angle assumptions of the vehicle it can be assumed that there is a relation between the rotational angles in body frame and the rotational velocity in world frame which is $\dot{\phi} \approx \mathrm{p}$, $\dot{\theta} \approx \mathrm{q}$ and $\dot{\psi} \approx \mathrm{r}$ [47]. This assumptions makes it possible to rewrite equation (3.24) which becomes

$$\ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{u_2}{I_{xx}}, \tag{3.25}$$

$$\ddot{\theta} = \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} + \frac{u_3}{I_{yy}}, \tag{3.26}$$

$$\ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{u_4}{I_{zz}}. \tag{3.27}$$

With equation (3.21), (3.22), (3.23), (3.25), (3.26) and (3.27), which describe the translational and rotational motion of the quadrotor, the complete model can be built, which looks as follows

$$\ddot{\mathrm{x}}_{\mathcal{A}} = \frac{(\mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\psi)u_1}{m_q}, \tag{3.28}$$

$$\ddot{\mathrm{y}}_{\mathcal{A}} = \frac{(\mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{s}\phi\mathrm{c}\psi)u_1}{m_q}, \tag{3.29}$$

$$\ddot{\mathrm{z}}_{\mathcal{A}} = \frac{(\mathrm{c}\phi\mathrm{c}\theta)u_1}{m_q} - g, \tag{3.30}$$

$$\ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{u_2}{I_{xx}}, \tag{3.31}$$

$$\ddot{\theta} = \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} + \frac{u_3}{I_{yy}}, \tag{3.32}$$

$$\ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{u_4}{I_{zz}}. \tag{3.33}$$

The collective thrust is described by u_1 and the roll, pitch and yaw moments are represented by u_2, u_3 and u_4, respectively. The under-actuated system has can move in 6 degrees of freedom with 4 control inputs.

3.2.2 Linearized Model

For linear control it is important to bring the model into state space representation. First the system has to be linearized around the hovering point, which means that $u_1 = mg$. The states for the linearized model are chosen as follow $x = [\mathrm{x}_{\mathcal{A}}\ \dot{\mathrm{x}}_{\mathcal{A}}\ \mathrm{y}_{\mathcal{A}}\ \dot{\mathrm{y}}_{\mathcal{A}}\ \mathrm{z}_{\mathcal{A}}\ \dot{\mathrm{z}}_{\mathcal{A}}\ \phi\ \dot{\phi}\ \theta\ \dot{\theta}\ \psi\ \dot{\psi}]^T$. The linearized state space representation follows the common model, which looks as follows.

$$\dot{x} = \mathbf{A}x + \mathbf{B}u, \tag{3.34}$$

$$y = \mathbf{C}x + \mathbf{D}u, \tag{3.35}$$

where $\mathbf{A}$ is the state matrix, $\mathbf{B}$ the input matrix, $\mathbf{C}$ the output matrix and $\mathbf{D}$ the feed-through matrix. The linear system has the advantage that simple and systematic controllers can be applied which still give excellent outputs. The following equilibrium points have been chosen

$$
\begin{aligned}
x_1 &= \mathrm{x}_{\mathcal{A}} = 0 & x_5 &= \mathrm{z}_{\mathcal{A}} = 0 & x_9 &= \theta = 0 & u_1 &= mg \\
x_2 &= \dot{\mathrm{x}}_{\mathcal{A}} = 0 & x_6 &= \dot{\mathrm{z}}_{\mathcal{A}} = 0 & x_{10} &= \dot{\theta} = 0 & u_2 &= 0 \\
x_3 &= \mathrm{y}_{\mathcal{A}} = 0 & x_7 &= \phi = 0 & x_{11} &= \psi = 0 & u_3 &= 0 \\
x_4 &= \dot{\mathrm{y}}_{\mathcal{A}} = 0 & x_8 &= \dot{\phi} = 0 & x_{12} &= \dot{\psi} = 0 & u_4 &= 0.
\end{aligned}
$$

The equilibrium points secure that the system will operate reasonably within a certain operating range around them. It has to be kept in mind that the system will behave as simulated only in a certain range around these points. Applying these equilibrium points to the linearized matrices gives the following matrices

$$
\mathbf{A} = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -g & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & g & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \tag{3.36}
$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{m_q} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_{xx}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_{yy}} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{I_{zz}} \end{bmatrix}, \tag{3.37}$$

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.38}$$

and

$$\mathbf{D} = 0. \tag{3.39}$$

The inertias of the quadrotor are specific to the model and can easily be applied to the matrices. The mass of the quadrotor is also model specific. The earth gravity constant g is used which is -9.81$\frac{m}{s^2}$.

3.3 System Architecture

In an actual system, the implementation looks a little bit different than the model. Unless direct motor control is implemented, it is usual to use a cascade control architecture. Normally a LL controller is controlling the attitude of the quadrotor and a HL controller is implementing position, velocity, acceleration or similar control that fits the specific needs. A generic control structure for the system of this thesis can be seen in Figure 3.2.

The two different systems have their own FC, which can be interfaced with the HL control implemented in this thesis. For the AscTec the LL controller running the attitude and altitude control loop is the AscTec AutoPilot. For the QAV250 frame base quadrotor, the PixRacer acts as attitude controller. Both come with their own advantages which are mainly the attitude control loop speed of 1kHz on the AscTec and the flexibility to use electromechanical components for the PixRacer. The attitude controller is utilized to stabilize the quadrotor. Both FC are interfaced by the Odroid XU4, which is the HL controller. The HL side of the control is completely implemented using ROS.

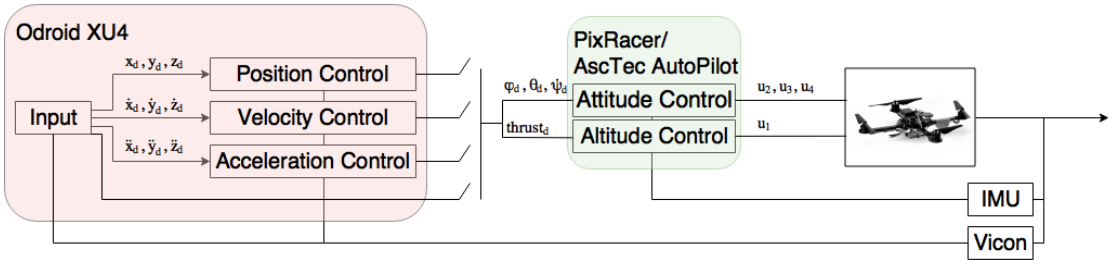


Figure 3.2: Overview of the closed loop control architecture implemented and parts where the different controllers are running.

The implemented architecture makes it possible to generate an input signal onboard the quadrotor, which can directly send its generated desired inputs to the `asctec_mav_framework` or the `mavros` framework on the different UAV's. These

desired inputs could be positions, velocities and/or accelerations. The respective HL controllers will ensure that these points are met and send roll and pitch angles, heading velocity and thrust values to the LL control. Additionally, it is possible to bypass these HL control algorithms and implement your own which sends directly desired roll and pitch angles, yaw velocity and a normalized thrust value. The attitude controller converts these desired orientation inputs into control inputs for the motors. Additionally, the altitude controller is converting the desired thrust in the collective thrust input into the motors. The motor inputs are sent to the motors and converted to thrust and momentum created by each individual motor. The motor inputs with specific dynamics are producing the outputs of the system that will be captured using sensors. The Vicon sensor will be used as a feedback for the HL side of the controls. It gives feedback about the position and orientation of the object created. This feedback can be used to estimate the velocity and accelerations of each state. It can be said that the Vicon sensor can easily be replaced by GPS or a camera using simultaneous localization and mapping algorithms. IMU is used for the attitude control. It feeds the rotational velocity and linear acceleration back to the PixRacer or AscTec AutoPilot. The inputs to the FL controllers could come from mapping, trajectory generation, obstacle avoidance, etc. algorithms. Additionally it is possible to add more sensors like Sonar, IR, laser, etc. for specific applications.

Chapter 4

Applications

The main purpose for this framework was to be as flexible as possible for different applications. The ROS framework has to be designed to work with multiple heterogeneous robots and communicate with them. The two main projects that the quadrotors are used for are the Micro Autonomous Systems and Technology(MAST) and Aerial Suppression of Airborne Platforms (ASAP) project implemented by the MARHES laboratory. The focus and implementation of these projects will be discussed.

4.1 MAST

4.1.1 Introduction

For the MAST project, the MARHES lab is developing a system which is able to effectively deploy and coordinate a network of heterogeneous robots, as seen in Figure 4.1. The idea is to be able to use small and cheap universal ground robots that map specific information of an area and send it via an optical link to aerial vehicles. The

aerial vehicles can sufficiently get information from multiple ground robots and bring it to a home station which would act as a cloud. Using the strength of node of the whole system makes it possible to overcome limitations of specific robots. The cloud is then powerful enough to use this information and extract what it needs. Such a framework would be helpful in situations like surveillance of complex environments, search and rescue, localization of targets, etc. The advantage of having an optical link between the ground robots and aerial robots is that it has a robust high communication rate. An optical link can only be jammed if the intruder is physically between the transmitter and receiver, which makes it interesting for security applications. As ground robots, the Kamigami Dash robots have been modified so that they have ROS on-board a Raspberry Pi zero. They are equipped with cameras and will take pictures of the environment. When their internal storage is full the quadrotor will fly above them and extract the data. The quadrotor will then fly to a ground station, which will represent the cloud and upload the data, so that it will be able to go back and get more information as soon as it has uploaded all data to the ground station. For this application, the quadrotor has to be able to do waypoint navigation and hover above an object to transmit and receive data. The quadrotor used for this application is the AscTec Hummingbird due to its advantage of superior flight time of up to 20 minutes. First the waypoint architecture will be discussed. Then a velocity estimator for the controller is implemented and tested before the results of the implemented linear control are presented.

4.1.2 Architecture

For the waypoint navigation, a position control which sends roll and pitch angles, yaw velocity and a normalized thrust reference directly to the LL controller has been developed. It is a PID controller and the schematic can be seen in Figure 4.2. The Odroid XU4 is still used as microprocessor running the HL controller with the

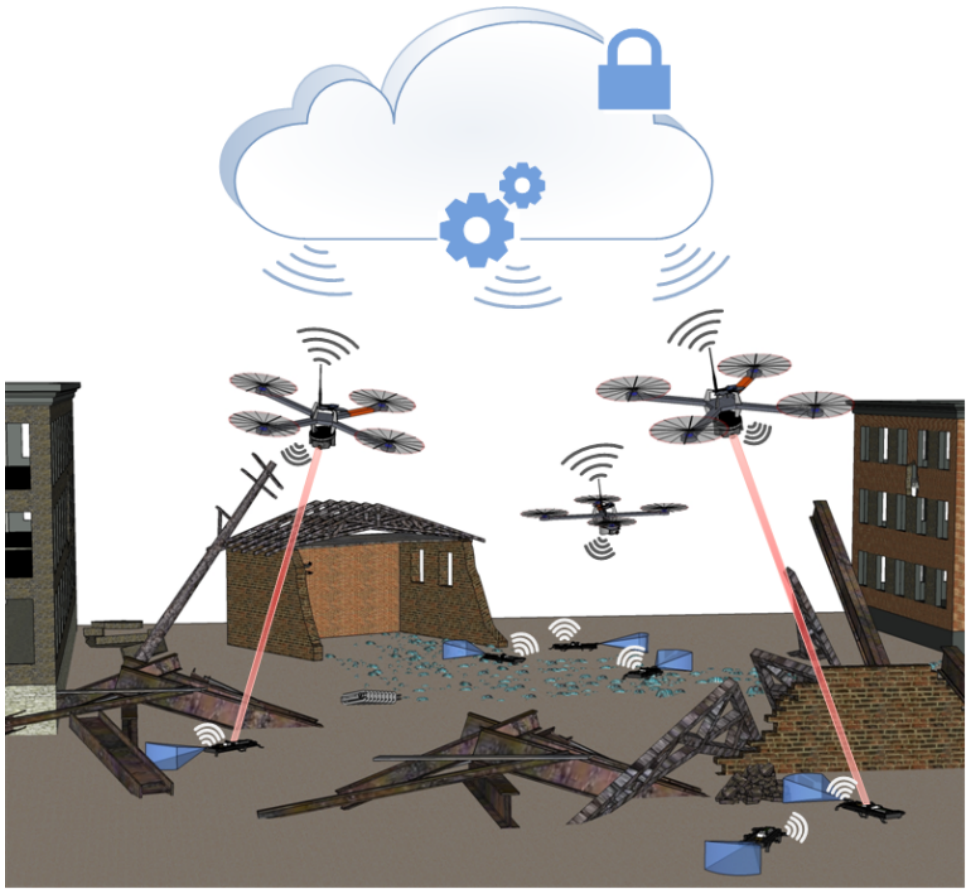


Figure 4.1: Heterogeneous robot system consisting of aerial and ground units. Cloud resources to optimize the mission.

`asctec_mav_framework` ROS package. The desired x and y values are given to the PID controller, which sends a reference to the transform node. The transform block will, depending on the heading, send the desired roll and pitch angles to the FC. The desired heading is sent to the PID algorithm and a reference heading velocity will be sent to the attitude control. The height control is controlled similarly, however the desired thrust value will be a normalized number between 0 an 1, which is converted by the LL altitude control in collective thrust created by the quadrotor. The reason to bypass the internal position control provided by `asctec_mav_framework` is that the provided position control was not agile and precise enough. It is designed for outdoor

environments with GPS feedback, so the focus was on robustness and reliability. Using Vicon as position feedback gives less noise with higher precision.

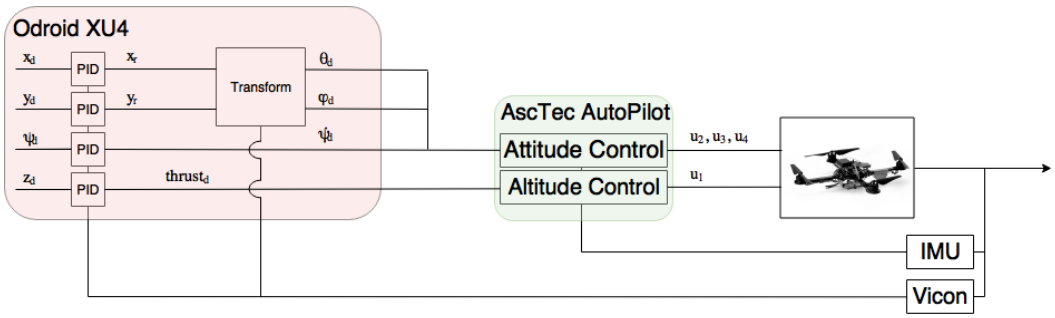


Figure 4.2: Control schematic for waypoint following of AscTec Hummingbird. The `asctec_mav_framework` controllers are bypassed and only used to send angles and thrust references to the LL attitude controller.

The PID controllers for each state look as follows

$$x_r = K_{px}e_x(t) + K_{ix}\int_0^t e_x(\tau)d\tau + K_{dx}\frac{de_x(t)}{dt}, \tag{4.1}$$

$$y_r = K_{py}e_y(t) + K_{ix}\int_0^t e_y(\tau)d\tau + K_{dx}\frac{de_y(t)}{dt}, \tag{4.2}$$

$$\dot{\psi}_d = K_{p\psi}e_\psi(t) + K_{i\psi}\int_0^t e_\psi(\tau)d\tau + K_{d\psi}\frac{de_\psi(t)}{dt}, \tag{4.3}$$

$$thrust_d = K_{pz}e_z(t) + K_{iz}\int_0^t e_z(\tau)d\tau + K_{dz}\frac{de_z(t)}{dt}, \tag{4.4}$$

,where x_r, y_r, $\dot{\psi}$ and $thrust_d$ are the inputs into the system. x_r and y_r are a reference that goes into the transform, and the transform calculates the desired roll and pitch that go to the LL controller. K_p, K_i and K_d are the tunable control gains. The error is shown by e.

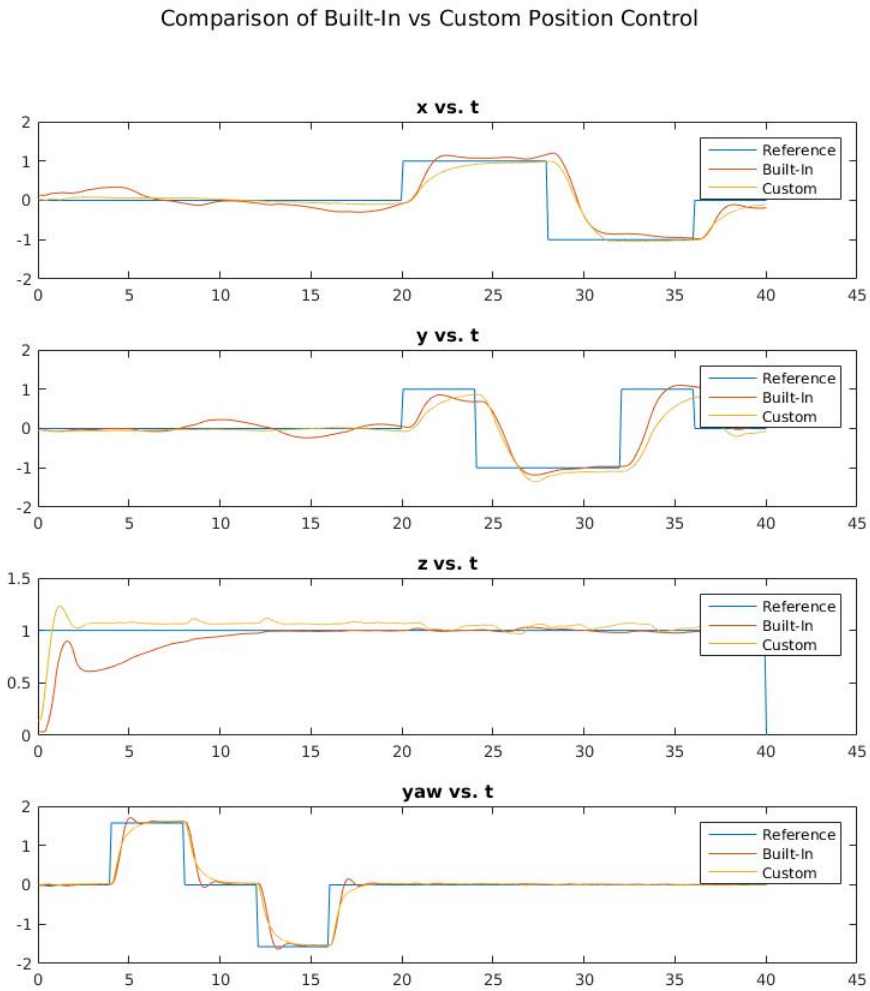


Figure 4.3: Waypoint following results for AscTec Hummingbird on-board build in controller vs. custom linear controller.

4.1.3 Velocity Estimator

The velocity estimator has been used for the derivative part of the PID control algorithm. Different methods have been taken into account which are the Two-Point Derivative, Three-Point Derivative and Al-Alaoui Derivative [48]. These derivatives

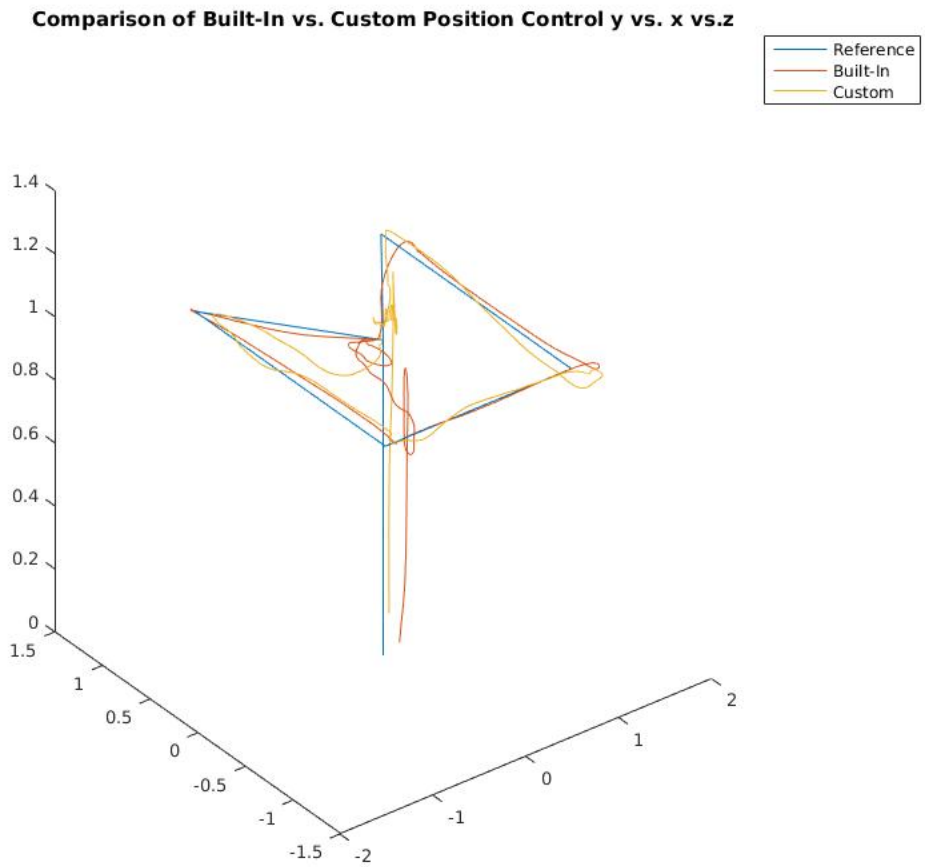


Figure 4.4: Waypoint following results for AscTec Hummingbird on-board build in controller vs. custom linear controller.

can be calculated as follows

$$\text{Two-Point} : v_x = \frac{x(k) - x(k-1)}{T_s}, \tag{4.5}$$

$$\text{Three-Point} : v_x = \frac{x(k+1) - x(k-1)}{2T_s}, \tag{4.6}$$

$$\text{Al-Alaoui} : v_x = -\frac{1}{7}v_x(k-1) + \frac{8(x(k) - x(k-1))}{7T_s}, \tag{4.7}$$

where x and v_x are the position and velocity, respectively. It can be noticed that the two point estimator is a linear estimation between two points, this works very well if there is no noise in the system. Even though the Vicon system provides reliable feedback, a small amount of noise is amplified in the velocity. This can lead to problems, especially with small step sizes. The three point derivative is less affected by noise due to the fact that it is taking linear measurements of 2 time steps. The real-time implementation problem is that it requires a future measurement. The Al-Alaoui velocity estimation is taking the previous measurement into account. The comparison of the estimators in the real implementation can be seen in Figure 4.5.

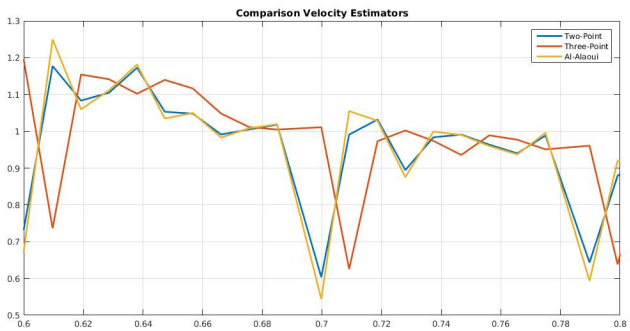


Figure 4.5: Comparison of the velocity estimator algorithms explained.

It can be seen that the three-point estimator is ahead by one time step. This is due to the fact that the estimation takes future measurements into account. The two-point and Al-Alaoui estimator perform similarly, whereas it can be noted that the spikes of the Al-Alaoui algorithm has higher peaks. Taking this into account, it has been decided that the Two-Point estimation is the best one for the real-time implementation and will be used for the PID position control.

4.1.4 Demo

As explained before, there will be one or more quadrotors in the demo. The UAV's are supposed to transport information from ground robots to a ground station. The quadrotor performance has been tested and the ROS framework has been implemented and tested. To be able to go from agent to agent, the UAV gets the position feedback of the ground robots. The results of the test can be seen in Figure 4.6. It shows the quadcopter switching between ground agents. In the blue region it is

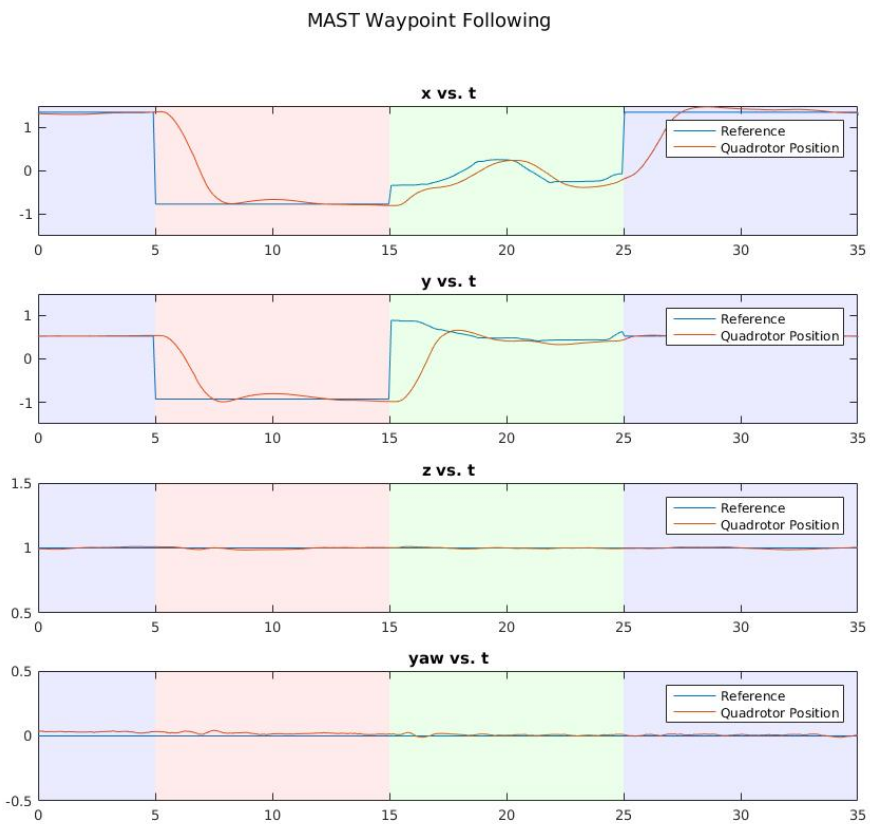


Figure 4.6: Quadcopter approaching different ground vehicles. In the blue region it tries to hover over the base station, in the red region it follows a stationary Kamigami Dash robot, and in the green region it tries to follow a moving Kamigami Dash robot.

trying to hover over the ground station, to get information from the Kamigami Dash robots. Then it moves towards the first stationary Kamigami, which is shown by the red region. When it received all the information it needed it goes to the next Kamigami, which is shown by the green region. This Kamigami is moving and the quadrotor is following it. When it receives all of the information it needs from the ground robot it flies back to hover over the ground station, where it can upload all the data it has collected. It can be seen that the quadrotor is able to switch between ground units, and is able to hold position reliably if the ground robot is not moving. If the ground robot is moving, it hovers above it with some position latency, however it is still performing well enough to be in the region of getting information. The ROS framework on each node in the system made it simple to communicate between robots and get information about their status.

4.2 ASAP

4.2.1 Introduction

Due to the increase of personal and commercial aerial vehicles in the last few years, more research is being conducted on security in possible threat situations. This includes detection, tracking, and neutralization. The research proposed here works towards a threat capture system. First, when a threat is detected, forward stochastic reachability analysis is used to evaluate the exact future probability distribution of the threat from its current location. This is followed by an optimal control problem to optimize the capture probability [21]. This novel approach has been developed and tested in the MARHES lab. The quadrotors used for this experiment is the QAV250 with on-board microprocessor.

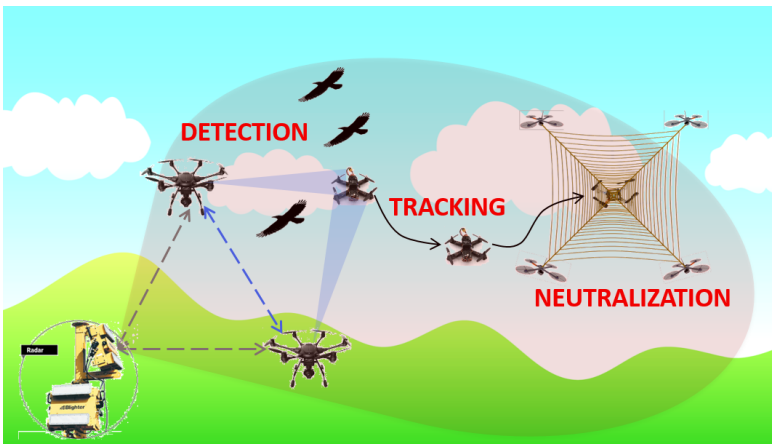


Figure 4.7: ASAP project. The idea of vehicle detection, tracking and neutralization.

4.3 System Schematic

The control architecture for this project makes use of the implemented position controller included with the ROS packages. This makes the implementation simple and saves time. Since the reachability analysis is running in Matlab and it needs Matlab functions, the implementation makes use of the new developed Matlab Robotics Toolbox. This toolbox makes it possible to receive and send ROS messages that can be integrated in the framework. The system schematic can be seen in Figure 4.8. Matlab is calculating way-points for the pursuer and threat. The `polyfit` function is computing trajectory coefficients for a third order polynomial that are sent to the quadrotors. The quadrotors execute these trajectories on-board and are able to switch their trajectories for a receding horizon environment in which updated coefficients get sent continuously.

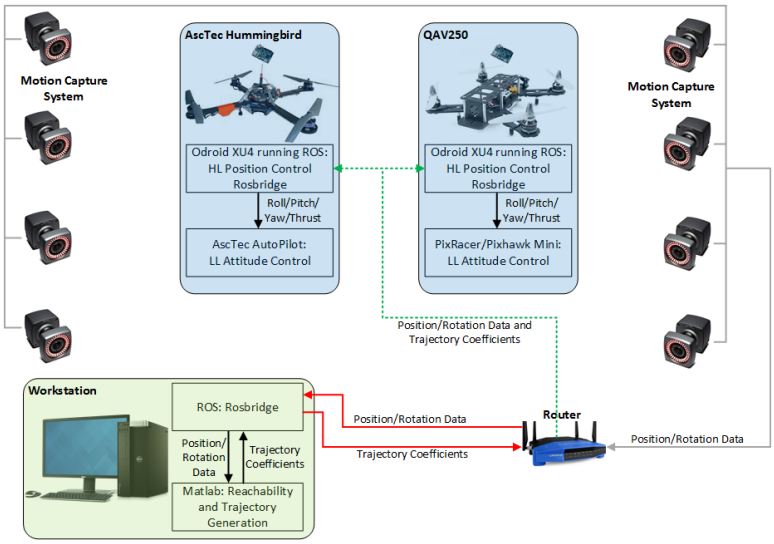


Figure 4.8: System overview of the implementation.

4.4 Trajectory Tracking

For the implementation of the project, a trajectory generation based on 3rd order polynomials has been implemented. This guarantees smooth path generation of waypoints. The coefficients are sent to the quadrotor. The quadrotor is able to execute the trajectory with respect to time. The $\mathrm{x}_{\mathcal{A}}$, $\mathrm{y}_{\mathcal{A}}$ and $\mathrm{z}_{\mathcal{A}}$ positions are calculated as follows

$$\mathrm{x}_{\mathcal{A}} = a_{x0} + a_{x1}t_d + a_{x2}t_d + a_{x3}t_d, \tag{4.8}$$

$$\mathrm{y}_{\mathcal{A}} = a_{y0} + a_{y1}t_d + a_{y2}t_d + a_{y3}t_d, \tag{4.9}$$

$$\mathrm{z}_{\mathcal{A}} = a_{z0} + a_{z1}t_d + a_{z2}t_d + a_{z3}t_d, \tag{4.10}$$

where a_{x0}, a_1, a_{x2} and a_{x3} are the polynomial coefficients to calculate the position in inertial frame. For $\mathrm{y}_{\mathcal{A}}$, x is switched with y and for $\mathrm{z}_{\mathcal{A}}$, x is switched with z. The time duration is represented by t_d. To test the performance, a figure eight has been flown. To generate the figure eight, a different control law for the position generation

has been used which is able to test the system with a difficult trajectory and can be simplified by changing the gain k. It look as follows

$$\mathrm{x}_{\mathcal{A}} = sin(kt_d) \tag{4.11}$$

$$\mathrm{y}_{\mathcal{A}} = sin(0.5kt_d) \tag{4.12}$$

$$\mathrm{z}_{\mathcal{A}} = 0.2sin(0.25kt_d). \tag{4.13}$$

The results of the test with different gains can be seen in Figure 4.9. This shows a 2D view from above. It can be seen that at a slow speed the quadrotor is following the trajectory precisely. For k=1.5 and k =2.0 the quadcopter is overshooting, however it is still able to follow the trajectory. More detailed results of the tests can be seen in Appendix A. This figure shows each position with respect to time. There it can be seen that some delay exists in the trajectory following. The position z of the quadrotor is a little jittery, however it is still able to follow the trajectory. These tests give promising results and show that the quadrotor is able to follow generated positions.

4.5 Experimental Results

The framework has been implemented in the MARHES laboratory. It is an open-loop experiment. The reachability analysis is generating trajectories at the beginning of the experiment. These trajectory coefficients are sent to the quadrotor which also triggers a flag that initiates the trajectory execution loop. With the help of a duration timer, the waypoints along the path are generated at 100Hz and send to the on-board position controller of the UAV. Two different scenarios have been tested and evaluated. Their positions have been saved on-board the quadrotor to generate the results seen in Figure 4.10 and Figure 4.11. The threat can be seen in red. It is approaching the asset. The pursuer is following the generated trajectory that was

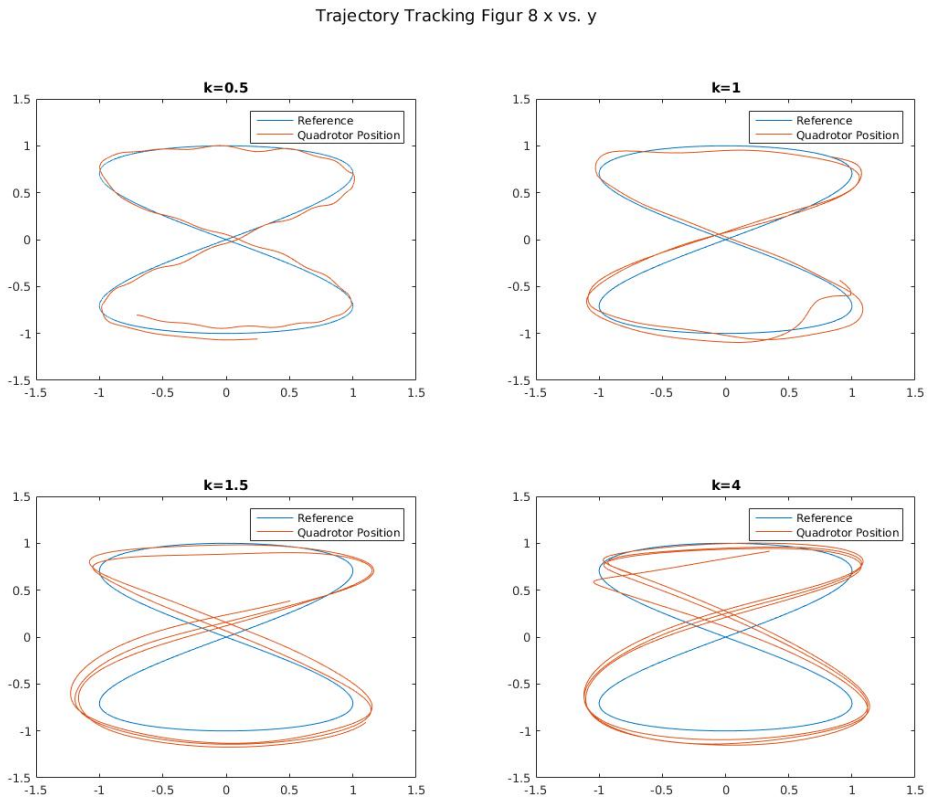


Figure 4.9: x vs. y graphs of figure eight trajectory tracking at different speeds

generated by the reachability analysis. It captures the threat both times before the threat can reach the asset. When the pursuer is inside the capture region of the threat, the quadrotors stop and the test is counted as successful. The results of the reachability analysis look promising.

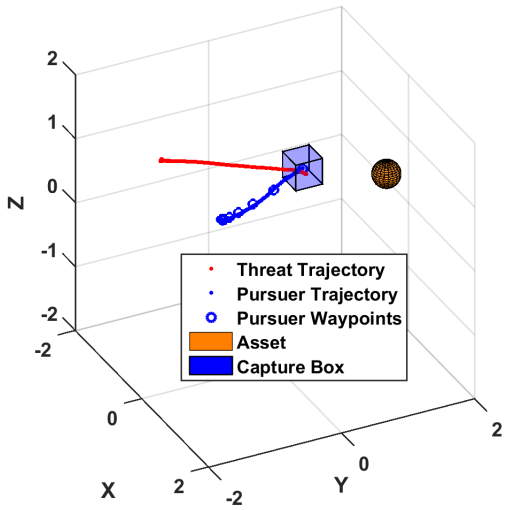


Figure 4.10: Experimental results of Scenario 1 for the implementation of the reachability analysis

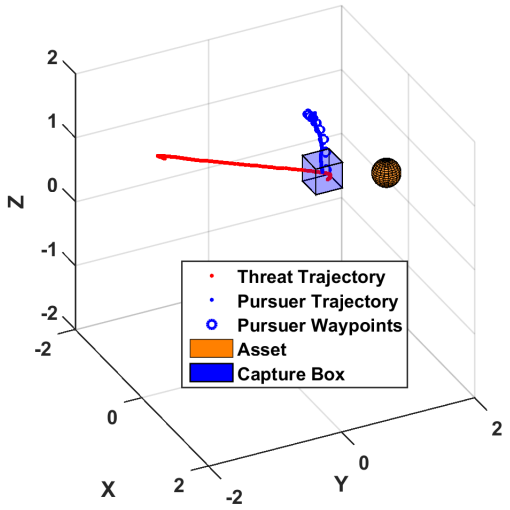


Figure 4.11: Experimental results of Scenarios 2 for the implementation of the reachability analysis

Chapter 5

Inverted Pendulum on a Quadrotor

As part of this thesis a framework for an off-centered inverted pendulum on a quadrotor has been developed. The goal is to have a quadrotor which can swing-up and balance the pendulum in a linear and rotational fashion. A model has been derived using Lagrangian. It adds the pendulum state to the complete nonlinear model representation derived in chapter 3 on page 21. The control design is presented and discussed. The controllers for each state of the hybrid systems have been developed and stability of the system has been tested using Lyapunov. After achieving the LQR gain matrix for the linear and rotational inverted pendulum, as well as the swing reduction gain matrix K_{swing} using Matlab, the system has been simulated. First the balancing maneuver has been simulated in Matlab. The full gazebo implementation has been developed and shows promising results. First steps of the real system implementation with preliminary results are shown.

5.1 Model

The model is following similar notation and assumptions as the one presented in chapter 3 on page 21. The schematics of the quadrotor model with attached pendulum can be seen in Figure 5.1.

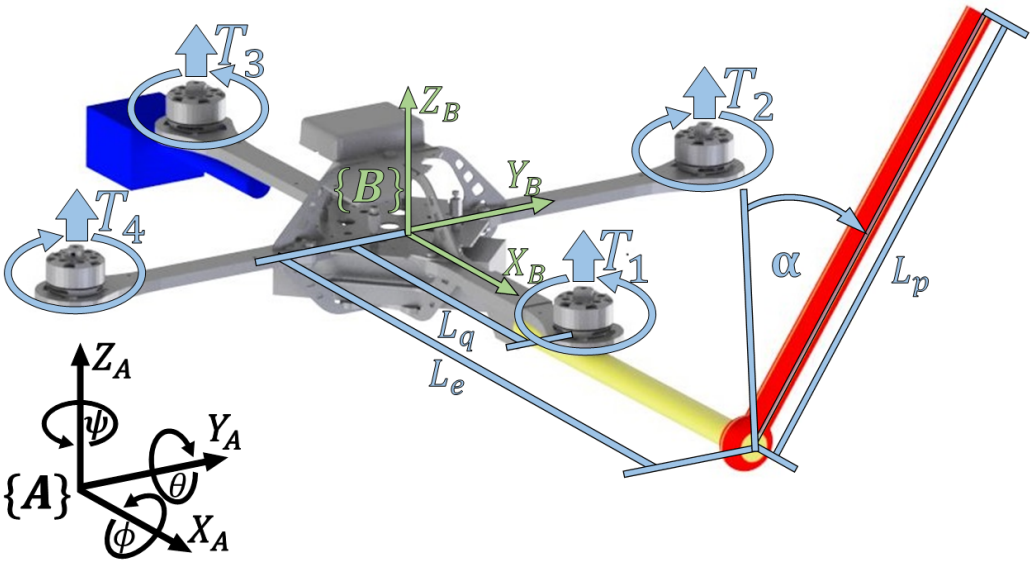


Figure 5.1: Schematic diagram of the quadrotor with the pendulum (red), pendulum arm extension (Yellow) and counterweight (blue) attached to it. The world frame $\{\mathcal{A}\}$ shown in gray and the body frame $\{\mathcal{B}\}$ shown in green with their related axes. The distance from the origin of the world frame to the origin of the body frame is shown in red. The system characteristics are symbolized in light blue.

In addition to the original picture, the attached pendulum is described by L_e, which is the length of the center of mass of the quadrotor to the pivot point where the pendulum is attached. The one degree of freedom of the pendulum is described by α, which rotates around $x_{\mathcal{B}}$. The pendulum angle is zero at the upward vertical position. The pendulum length is described by L_p. A counterweight is added to balance the additional weight by the pendulum and erase the moment created around the COM. Additionally, it allows for the assumption that the COM is not switching

due to the pendulum. The mass of the pendulum attachment is given by $m_p$and the counterweight mass is described by m_c.

The Lagrangian method is used to derive the model. The position of the pendulum is given by $\mathbf{x}_L = \begin{bmatrix} \mathrm{x}_L & \mathrm{y}_L & \mathrm{z}_L \end{bmatrix}^T$, respectively. Additionally the angular velocity vector in body frame is described by $\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$.

As seen in the Figure 5.1 the pendulum is off-centered in the x-axis of the quadrotor body frame, so that the suspension point of the pivot can be described as $(0, 0, L_e)$ in $\{\mathcal{B}\}$ [49]. The center of mass of the pendulum will be given by $\frac{1}{2}L_p$. So let us define $\mathbf{l}_e = \begin{bmatrix} 0 & 0 & L_e \end{bmatrix}$ and $\mathbf{l}_p = \begin{bmatrix} 0 & 0 & \frac{1}{2}L_p \end{bmatrix}$.

Hence, $\mathbf{q}$ and can be defined as,

$$\mathbf{q} = \begin{bmatrix} \mathrm{x}_{\mathcal{A}} & \mathrm{y}_{\mathcal{A}} & \mathrm{z}_{\mathcal{A}} & \phi & \theta & \psi & \alpha \end{bmatrix}. \tag{5.1}$$

Furthermore, it is possible to describe the position of the COM of the pendulum in the world frame by

$$\mathbf{x}_L = \boldsymbol{\xi} + \mathbf{R}_z(\psi)\mathbf{R}_y(\theta){\mathbf{l_e}}^T + \mathbf{R}_{S^-}^{S^+}\mathbf{R}_x(\alpha)\mathbf{l}_p^T. \tag{5.2}$$

The pendulum angle is defined in an inverted configuration of the suspension point reference frame, which is represented by $\{S^-\}$. Whereas $\{S^+\}$ reference frame is always parallel to $\{\mathcal{B}\}$. The corresponding matrix is given by

$$\mathbf{R}_{S^-}^{S^+} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{5.3}$$

The quadrotor position in world frame is defined by $\boldsymbol{\xi}$ and the pendulum positions also in world frame are given by $\mathbf{x}_L$. The angular velocities of the quadrotor $\boldsymbol{\omega}$ and

the pendulum angle $\dot{\alpha}$ are given in body reference frame and $\dot{\alpha}$ is assumed to be in the suspension point frame $\{S^-\}$.

$$\begin{bmatrix} \mathrm{x}_L \\ \mathrm{y}_L \\ \mathrm{z}_L \end{bmatrix} = \begin{bmatrix} \mathrm{x}_{\mathcal{A}} \\ \mathrm{y}_{\mathcal{A}} \\ \mathrm{z}_{\mathcal{A}} \end{bmatrix} + \begin{bmatrix} \mathrm{c}\theta\mathrm{s}\psi & -\mathrm{s}\psi & \mathrm{c}\psi\mathrm{s}\theta \\ \mathrm{s}\psi\mathrm{c}\theta & \mathrm{c}\psi & \mathrm{s}\psi\mathrm{s}\theta \\ -\mathrm{s}\theta & 0 & \mathrm{c}\theta \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{1}{2}L_p \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\mathrm{c}\alpha & \mathrm{s}\alpha \\ 0 & -\mathrm{s}\alpha & -\mathrm{c}\alpha \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ L_e \end{bmatrix}. \quad (5.4)$$

Equation (5.4) can be computed and written in full form as follows

$$\begin{bmatrix} \dot{\mathrm{x}}_{\mathcal{A}} \\ \dot{\mathrm{y}}_{\mathcal{A}} \\ \dot{\mathrm{z}}_{\mathcal{A}} \\ \omega_x \\ \omega_y \\ \omega_z \\ \dot{\mathrm{x}}_L \\ \dot{\mathrm{y}}_L \\ \dot{\mathrm{z}}_L \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -\mathrm{s}\theta & 0 \\ 0 & 0 & 0 & 0 & \mathrm{c}\phi & \mathrm{s}\phi\mathrm{c}\theta & 0 \\ 0 & 0 & 0 & 0 & -\mathrm{s}\phi & \mathrm{c}\phi\mathrm{c}\theta & 0 \\ 1 & 0 & 0 & 0 & L_e\mathrm{s}\theta\mathrm{c}\psi & -L_e\mathrm{c}\theta\mathrm{s}\psi & 0 \\ 0 & 1 & 0 & 0 & -L_e\mathrm{s}\theta\mathrm{s}\psi & L_e\mathrm{c}\theta\mathrm{c}\psi & \mathrm{c}\alpha \\ 0 & 0 & 1 & 0 & -L_e\mathrm{c}\alpha & 0 & \frac{1}{2}L_p\mathrm{s}\alpha \end{bmatrix} \begin{bmatrix} \dot{\mathrm{x}}_{\mathcal{A}} \\ \dot{\mathrm{y}}_{\mathcal{A}} \\ \dot{\mathrm{z}}_{\mathcal{A}} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\alpha} \end{bmatrix} \quad (5.5)$$

which can be presented by

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \quad (5.6)$$

For the Lagrangian method the potential and kinetic energy has to be calculated. The kinetic energy of the complete system can be expressed as follows,

$$T = \frac{1}{2}\dot{\mathbf{x}}_Q^T\mathbf{M}_Q\dot{\mathbf{x}}_Q + \frac{1}{2}\dot{\mathbf{x}}_L^T\mathbf{M}_L\dot{\mathbf{x}}_L + \frac{1}{2}\omega^T\mathbf{I}\omega + \frac{1}{2}I_P\dot{\alpha}^2, \quad (5.7)$$

or

$$T = \frac{1}{2}\dot{\mathbf{x}}^T\mathbf{M}\dot{\mathbf{x}} = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{J}(\mathbf{q})^T\mathbf{M}\mathbf{J}(\mathbf{q})\dot{\mathbf{q}} + \frac{1}{2}I_P\dot{\alpha}^2. \tag{5.8}$$

The matrix $\mathbf{M} = \text{diag}([m_q, m_q, m_q, I_{xx}, I_{yy}, I_{zz}, m_p, m_p, m_p])$. The mass of the quadrotor is represented by m_q and the pendulum mass is represented by m_p. The pendulum inertia is shown by I_P and the quadrotor inertia's are shown by I_{xx}, I_{yy} and I_{zz} about x, y and z axis.

Now that the kinetic energy is calculated, the potential energy calculation is the next step, It is given by V_p, which looks as follows

$$V_p = m_q g z_q + m_q g(z_Q - L\sin\theta - L\cos\alpha). \tag{5.9}$$

Since the kinetic and potential energy expressions have been calculated, it is possible to evaluate the Lagrangian,

$$\mathcal{L} = T - V_p. \tag{5.10}$$

The Euler-Lagrange equation can be used to calculated the dynamics of the off-centered pendulum attached to the quadrotor [50],

$$\frac{d}{dt}[\frac{\partial\mathcal{L}}{\partial\dot{\mathbf{q}}}] - \frac{\partial\mathcal{L}}{\partial\mathbf{q}} = \mathbf{f}_{ext}, \tag{5.11}$$

which can be written as,

$$\frac{d}{dt}[\mathbf{W}\dot{\mathbf{q}}] - \frac{1}{2}\frac{\partial}{\partial\mathbf{q}}[\dot{\mathbf{q}}^T\mathbf{W}]\dot{\mathbf{q}} + \frac{\partial V}{\partial\mathbf{q}} = \mathbf{f}_{ext}. \tag{5.12}$$

We obtain

$$\mathbf{W}\ddot{\mathbf{q}} + (\dot{\mathbf{W}} - \frac{1}{2}\frac{\partial}{\partial\mathbf{q}}[\dot{\mathbf{q}}^T\mathbf{W}])\dot{\mathbf{q}} + \frac{\partial V}{\partial\mathbf{q}} = \mathbf{f}_{ext}. \tag{5.13}$$

From Equation (5.13), one can obtain the mass, Coriolis and gravitational terms for the equation of motion of the flying inverted pendulum which looks as follows,

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \mathbf{f}_{ext}, \tag{5.14}$$

where $\mathbf{W} = \mathbf{J}^T(\mathbf{q})\mathbf{M}\mathbf{J}(\mathbf{q}) + \text{diag}([0, 0, 0, 0, 0, 0, 0, I_P])$.

With these equations it is possible to get the complete model of the system. Additionally, it has to be said that it is assumed that the pendulum mass is so small that it does not affect the quadrotor. The quadrotor complete nonlinear model can be written as

$$\ddot{\mathrm{x}}_{\mathcal{A}} = \frac{(\mathrm{c}\psi\mathrm{s}\theta\mathrm{c}\phi + \mathrm{s}\phi\mathrm{s}\psi)u_1}{m_q + m_p + m_c}, \tag{5.15}$$

$$\ddot{\mathrm{y}}_{\mathcal{A}} = \frac{(\mathrm{s}\psi\mathrm{s}\theta\mathrm{c}\phi - \mathrm{s}\phi\mathrm{c}\psi)u_1}{m_q + m_p + m_c}, \tag{5.16}$$

$$\ddot{\mathrm{z}}_{\mathcal{A}} = \frac{(\mathrm{c}\phi\mathrm{c}\theta)u_1}{m_q + m_p + m_c} - g, \tag{5.17}$$

$$\ddot{\phi} = \frac{I_{yy} - I_{zz}}{I_{xx}}\dot{\theta}\dot{\psi} + \frac{u_2}{I_{xx}}, \tag{5.18}$$

$$\ddot{\theta} = \frac{I_{zz} - I_{xx}}{I_{yy}}\dot{\phi}\dot{\psi} + \frac{u_3}{I_{yy}}, \tag{5.19}$$

$$\ddot{\psi} = \frac{I_{xx} - I_{yy}}{I_{zz}}\dot{\theta}\dot{\phi} + \frac{u_4}{I_{zz}}, \tag{5.20}$$

$$\ddot{\alpha} = \frac{\frac{1}{2}m_pL_pL_e\mathrm{c}\alpha\ddot{\psi} + \frac{1}{4}m_pL_p^2\mathrm{c}\alpha\mathrm{s}\alpha\dot{\psi}^2 + \frac{1}{2}m_pL_pg\mathrm{s}\alpha + \frac{1}{2}m_pL_p\ddot{\mathrm{y}}_{\mathcal{A}}\mathrm{c}\alpha}{I_p + \frac{1}{4}m_pL_p^2} - B_p\dot{\alpha}. \tag{5.21}$$

It should be noted that the pendulum state $\ddot{\alpha}$ is effected by two movements, the linear acceleration in y and the rotational motion of the quadrotor.

5.2 Control Design

For the control design of an inverted pendulum on a quadrotor, Energy control will be used to swing-up the pendulum, and a LQR controller will be used to balance it. The decision between linear or rotational swing-up and balancing will be decided by a digital switch. Figure 5.2 shows the dual hybrid control design. For both cases, energy control will be used to get the pendulum out of its rest state and reach its upward vertical position with a zero velocity. This makes it easier for the LQR controller to take over and balance the pendulum in the upward vertical position. For the switch from energy control to LQR control some boundaries were defined. The controller switches to balance mode when α is less than ± 0.25 radians around the upward vertical position and has less than $0.4\frac{rad}{s}$ angular velocity. For safety reasons it is switching back if the angle of the pendulum is more than ± 0.35 radians from the upward vertical equilibrium point.

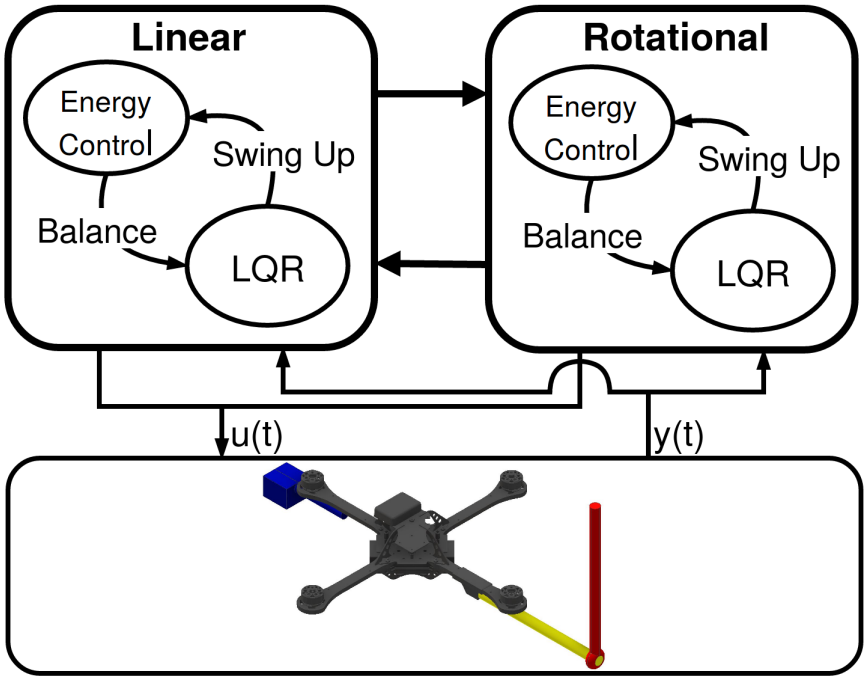


Figure 5.2: Block Diagram of the Control Schematic

5.2.1 Energy Control

The energy control that is described here is generating a rotational or linear acceleration, which will be fed into the low level controller. This means that in both cases the whole nonlinear model is considered. The energy control has been chosen due to its advantage that the upward vertical pendulum position is reached with zero rotational velocity, which makes the switch between controllers easier [51]. First, the potential and kinetic energy of the pendulum is calculated by

$$E = \frac{1}{2} I_p \dot{\alpha} + \frac{1}{2} m_p g L_p (\cos \alpha - 1). \tag{5.22}$$

The upward vertical position when $\alpha = 0$ has no energy, so that it can be said that the desired energy of the pendulum is 0. Different solutions and approaches have been tested, where the obvious approach is to swing it up as fast as possible, with the biggest control input possible. This comes with disadvantages like possible chattering. Therefore different control laws have been implemented, a common one that avoids chattering looks as follows

$$u = \text{sat}_{u_{max}}(k(E - E_o)\text{sign}(\dot{\alpha} \cos \alpha)). \tag{5.23}$$

A control saturation has been implemented by $\text{sat}_{u_{max}}$, which saturates u_{max}. Additionally, a control gain has been presented by k. Tuning this gain is up to the user and can be customized to the system. A faster switching of the control signal is achieved by $\text{sign}(\dot{\alpha} c \alpha))$. As explained before, the desired energy E_o is zero. The actual system energy E, which was calculated using Equation (5.22), will reach zero with this control law. In the downward vertical position, the control law will be zero due to $sign(\dot{\alpha} c \alpha)$, so that at the energy control input at this point is defined as $mg\frac{L_p}{2}$. All of these tweaks in the control law make it more adjustable in the real system, and the user has to set u_{max} and k carefully to get the best output. It is obvious

that bigger numbers bring the pendulum up faster, however there is the possibility that they make the whole system unstable.

5.2.2 LQR Control

In this section a brief introduction of LQR control is presented. The system needs to go to a specific set point based on a specific cost of the system. This cost could be time, energy, fuel capacity, etc. How do we determine an input $u(t)$ to make this happen? To minimize the quadratic cost which is J;

$$\mathbf{J} = \int_0^\infty (x^T \mathbf{Q} x + u^T \mathbf{R} u) dt \tag{5.24}$$

The solution to this equation is an LQR controller, a linear quadratic regulator.

$$u(t) = -\mathbf{K}x, \text{ where } \mathbf{K} = \mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} \tag{5.25}$$

That minimizes Equation (5.24) and P is the solution to the algebraic ricatti equation,

$$\mathbf{A}^T\mathbf{P} + \mathbf{P}\mathbf{A} - \mathbf{P}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{P} + \mathbf{Q} = 0 \tag{5.26}$$

Where the choice for $\mathbf{R}$ and $\mathbf{Q}$ are dependent on the goals of the minimization of the system.

LQR for Linear Balancing

For the linear LQR balancing control, the model will be reduced, due to the ability to decouple the states of the model, to 6 states which are $\alpha, \dot{\alpha}, \mathrm{y}_{\mathcal{A}}, \dot{\mathrm{y}_{\mathcal{A}}}, \phi, \dot{\phi}$ This makes

the control easier and clear. The remaining states will be handled by an individual position controller, so that they can be left out to simplify the calculation. The model was then linearized around the hovering position and the right constraints and equilibrium points were applied to the model. The assumption is applied that the position control of ψ is controlling the heading and it has no effect on the pendulum for this analysis, since it is suppose to be 0 at all times of the linear balancing. The linearized model looks as follows

$$\ddot{\mathrm{y}}_{\mathcal{A}} = \frac{-\phi u_1}{m_q + m_p + m_c}, \tag{5.27}$$

$$\ddot{\phi} = \frac{u_2}{I_{xx}}, \tag{5.28}$$

$$\ddot{\alpha} = \frac{\frac{1}{2}m_p L_p g\alpha + \frac{1}{2}m_p L_p \ddot{\mathrm{y}}_{\mathcal{A}}}{I_p + \frac{1}{4}m_p L_p^2}. \tag{5.29}$$

The goal is to balance the pendulum at its upward vertical position, so the equilibrium point for α has been chosen to be zero. To implement the LQR controller, the feedback gain matrix, $\mathbf{K}$, must be determined. To do this, the performance index matrix $\mathbf{R}$, and the state cost matrix $\mathbf{Q}$ must be positive definite. They were chosen as

$$\mathbf{Q} = 20(\mathbf{C}^{\mathrm{T}} * \mathbf{C}), \tag{5.30}$$

and

$$\mathbf{R} = p * [1], \tag{5.31}$$

where $p = 0.2$ which allows the system to reach the states faster without error. The states that are available in the real world implementation by the Vicon motion capture system are $\mathrm{y}_{\mathcal{A}}, \phi$ and α. Therefore it can be stated that $\mathbf{C} = [1,0,0,0,0,0;0,0,1,0,0,0;0,0,0,0,1,0]$. The matrix $\mathbf{K}$ will then be fed back into the system, so that $\dot{x} = (\mathbf{A} - \mathbf{BK})x$.

LQR for Rotational Balancing

Similarly to the linear balancing for the rotational balancing, the model will be reduced to 4 states which are $\alpha, \dot{\alpha}, \psi, \dot{\psi}$. This can be assumed since the goal is to balance α and the pendulum is not effected by the other states in the rotational setup. Therefore, an independent position controller will handle these states. The nonlinear system will be linearized. The characteristics of the system and the chosen equilibrium points have been applied to the linearized state space matrices. For the pendulum state it is assumed that the linear movement does not affect it. Since this study wants to balance the pendulum in the upward vertical position, it is important to choose zero as an equilibrium point for the angle of the pendulum. The linearized model looks as follows

$$\ddot{\psi} = \frac{u_4}{I_{zz}}, \tag{5.32}$$

$$\ddot{\alpha} = \frac{\frac{1}{2}m_p L_p L_e \ddot{\psi} + \frac{1}{2}m_p L_p g \alpha}{I_p + \frac{1}{4}m_p L_p^2}. \tag{5.33}$$

The other states want to be driven to zero. The final linear state space representation has been used to develop an LQR controller that will balance the pendulum for small angles around the upward vertical position. Similar to the linear balancing, the goal is to minimize the error and reach the desired states faster. The weights have been chosen as follows

$$\mathbf{Q} = 20(\mathbf{C}^{\mathrm{T}} * \mathbf{C}), \tag{5.34}$$

and

$$\mathbf{R} = p * [1], \tag{5.35}$$

where $p = 0.2$. The resulting static gain matrix $\mathbf{K}$ was fed back into the system.

5.3 Stability Analysis

Lyapunov function is used to determine the stability of the hybrid system. The system can be seen as a hybrid switch system, due to its switch from energy to LQR control [52]. The two states are used to achieve different goals, the energy control is suppose to bring up the pendulum to the upwards vertical position $\mathbf{r} = 0$ and the LQR controller is suppose to stabilize it. As explained before, it is assumed that the pendulum weight is so small that it does not affect the quadrotor. To analyze the stability we let V_1 and V_2 be Lyapunov function candidates for swinging-up and balancing, respectively.

$$V_1 = \frac{1}{2}E^2 + \delta, \qquad \forall\ \mathbf{r} \in \mathbb{R}^2 \tag{5.36}$$

$$V_2 = \mathbf{r}^T\mathbf{P}\mathbf{r}, \qquad \forall\ \mathbf{r} \in \Gamma_2 \tag{5.37}$$

where Γ_2 is the region in the neighborhood of the equilibrium point. $\mathbf{P}$ defines a positive-definite, symmetric matrix. The design parameter which can be varied is δ.

In the swing-up mode, $\mathbf{r} \in \Gamma_1 = \mathbb{R}^2 \backslash \Gamma_2$ where $\Gamma_1, \Gamma_2 \subset \mathbb{R}^2$ are the regions in state space. The switching occurs when $V_i(\mathbf{r}) - V_j(\mathbf{r}) = 0$, where Γi represent the current region of operation and $i \neq j$. It can be stated that equilibrium point is asymptotically stable if the $\min(V_1, V_2)$ Lyapunov switching sequence is used. This can also be seen in Figure 5.3.

It can be seen that both Lyapunov functions approach zero, which shows that the system is stable in the sense of Lyapunov. The energy control brings it close to the origin and then the LQR control can take over to get the system quicker to the origin and have a better performance.

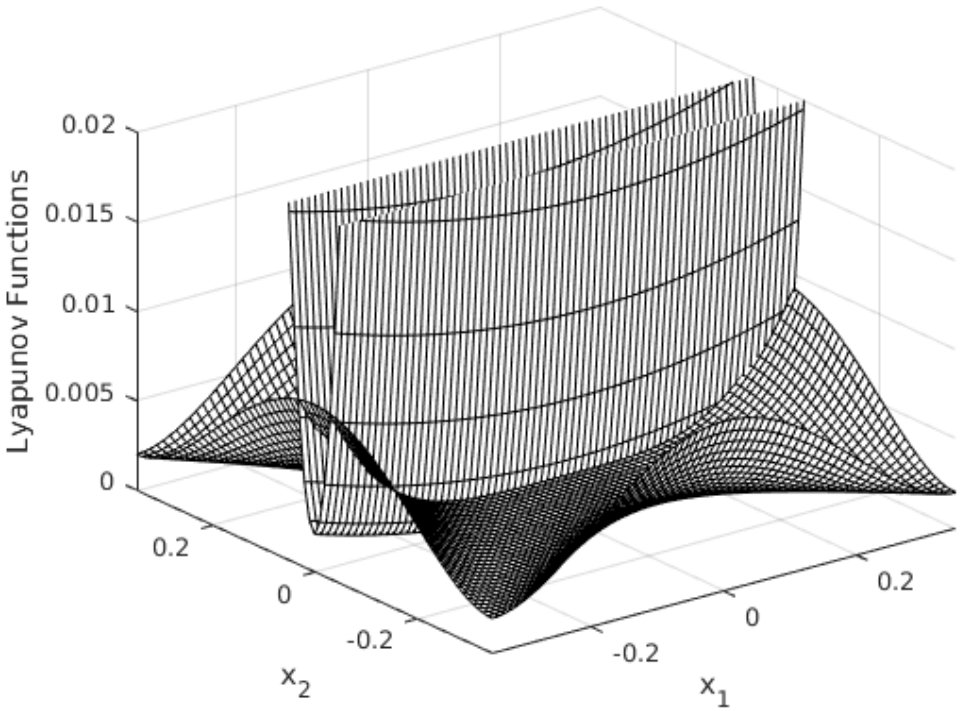


Figure 5.3: Stability analysis of the hybrid controller.

5.4 Matlab

Matlab has been used to simulate the LQR controller designed previously. The model used for the simulation is the one described in Chapter 3 on page 21. The gain matrices achieved in Section 5.2.2 on page 56 will be used to calculate $u = -\mathbf{K}x$ so that $\dot{x} = (\mathbf{A} - \mathbf{BK})x$. Additionally the swing reduction will be tested, so that it can be guaranteed the for the experiment. The pendulum will also be able to stabilize the downward vertical position. This adds stability to the system between the two modes, and makes it possible to guarantee that the swing up and balancing maneuvers can be performed from a resting pendulum.

These Matlab results will be essential for the next step, which is the Gazebo

simulation of the whole system. It will give an understanding of the transformation point, where the swing- up control can switch to balancing control. This will mostly depend on the angle of the pendulum. However, as an additional fact it will help that during the transformation the pendulum is already approaching the uprights vertical position, so that even though it might switch to the LQR when the pendulum has a greater angle away from the upward vertical position, its velocity is moving it already in the right direction.

5.4.1 Linear Pendulum

The simulation results for the linear balancing of the pendulum look promising and can be seen in Figure 5.4. The first graph shows the y position of the quadrotor compared to time, the second graph shows the pitch angle compared to time and the third one shows the pendulum angle compared to time. The system has been given some initial error, which is 0.4 m in the y position, 0.1 rad as pendulum angle and 0.1 $\frac{rad}{s}$ as pendulum angle velocity. It can be seen that the system is stable and approaches the desired state, which is zero for all states. The pendulum angle is stabilized first, at around 2 seconds together with the pitch angle of the quadrotor. The pendulum position has been given less weight in the LQR calculation, which shows in the result. The result displays this by first balancing the pendulum and then slowly approaching 0 as the desired y position.

The results for the linear balancing of the pendulum show that it is possible to balance it. Different runs have been made with different initial conditions to get a better understanding of the transformation point between the swing-up control and the balancing control, because that is the critical point of the actual implementation. Without hitting this point perfectly, the actual implementation will fail and the quadrotor will not be able to balance the pendulum. The tests have shown that the

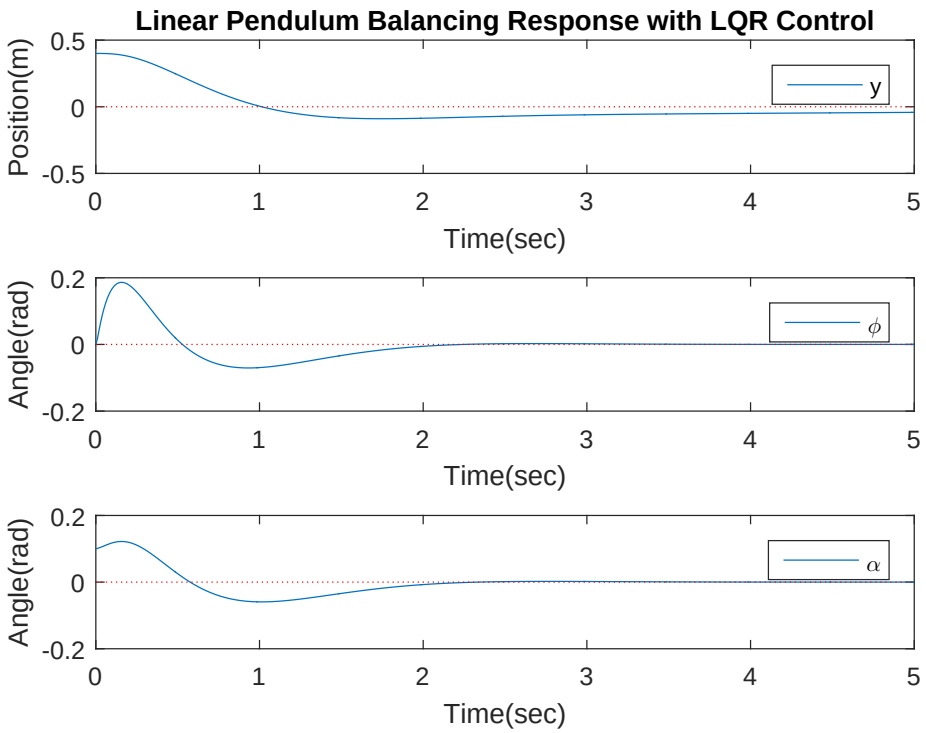


Figure 5.4: Responses of the states to balance the pendulum with LQR control. First graph shows the y position of the quadrotor, second graph shows the pitch angle of the quadrotor and the last graph shows the pendulum angle.

quadrotor is fairly comfortable balancing the pendulum when the angle is around ± 15 degrees.

5.4.2 Rotational Pendulum

The simulations for the rotary balancing of the pendulum about its upright vertical position can be seen in Figure 5.5. It can be seen that it is possible for the quadrotor to balance the pendulum. It also has to be considered that the heading is the weakest input into to the physical system due to the fact that the moment around z is created as a result of the moments of each propeller, whereas the moment around x and y of

the body is created as a result of differences in forces of two propellers. It can be seen that the pendulum can be balanced. The system has been given a initial condition of 0.19 rad for the yaw angle, 0.1 rad as pendulum angle and 0.1 $\frac{rad}{s}$ as pendulum angle velocity. The pendulum is balanced after approximately 1.3 seconds. The heading is moving aggressively to balance the pendulum and as soon as it is balanced, it is slowly moving to its desired value of 0 radians.

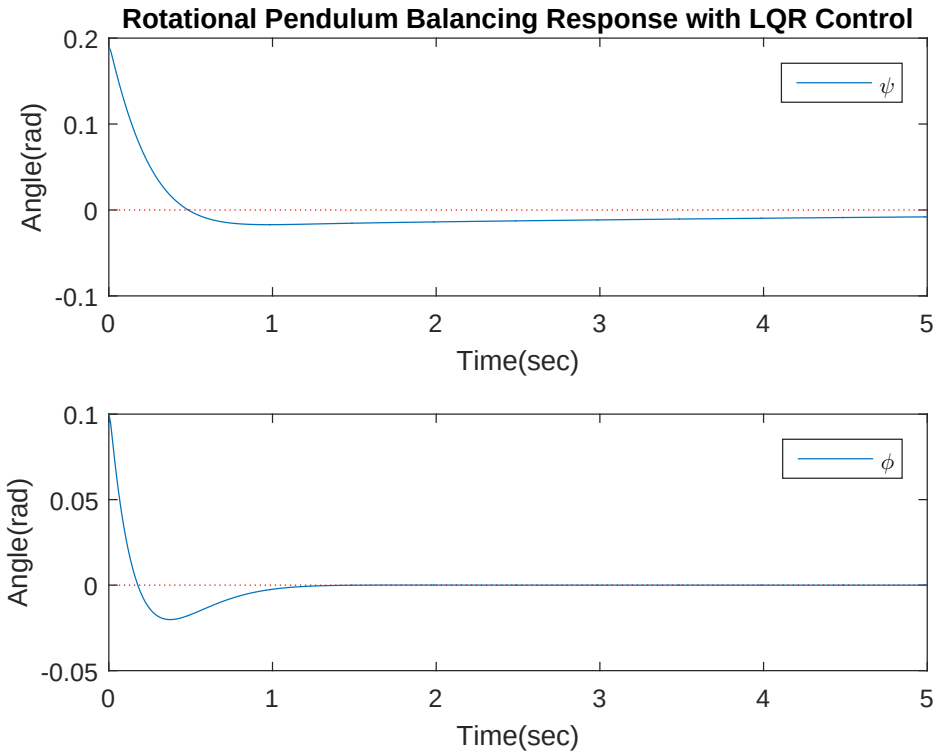


Figure 5.5: Responses of the states to balance the pendulum with heading change of the quadrotor. First graph shows the heading angle yaw of the quadrotor and the last graph shows the pendulum angle.

The transformation point between the swing-up and balancing control has been determined to be around ±10 degrees. The smaller range is due to the quadrotor dynamics, which show that heading control is weaker than the other control inputs. Multiple simulation runs with different initial conditions have been done to determine

this result. The results will be used as a reference for the actual system and further evaluated by the Gazebo simulation, which should give a deeper understanding of the real system.

5.4.3 Reduce Swinging

The swing reduction control is an important part for the actual system, however it is less important than the balancing. The positive part about the swing reduction is that gravity is helping balance the pendulum at its downward vertical position. The pendulum will naturally go towards the downward vertical position, however the quadrotor moves accordingly to reduce the swing of the pendulum, so that the actual system can switch between the two modes and start the swing-up from fresh, without having to wait until the pendulum does not swing anymore. The results of the swing reduction simulation with the help of an LQR controller can be seen in Figure 5.6. The responses shown were simulated with the initial condition of 0.1 m in the y position, 0.1 rad as pendulum angle and 0.1 $\frac{rad}{s}$ as pendulum angle velocity. It can be seen that the system is approaching its desired states in 2 seconds. First the pendulum angle and pitch angle of the quadrotor reach their desired state at around 1 second. The pendulum angle is approaching π, which is the downward vertical position of the quadrotor. After that, the quadrotor reaches the origin y position of 0 at around 2 seconds. For the swing reduction there has not been a determined point where the controller switches. The goal is to go from the balancing into the swing reduction, which can be done due to the fact that the pendulum would stabilize around π. The quadrotor is only used to reduce the swinging, acting as a damper and bring the pendulum faster to its vertical downward position.

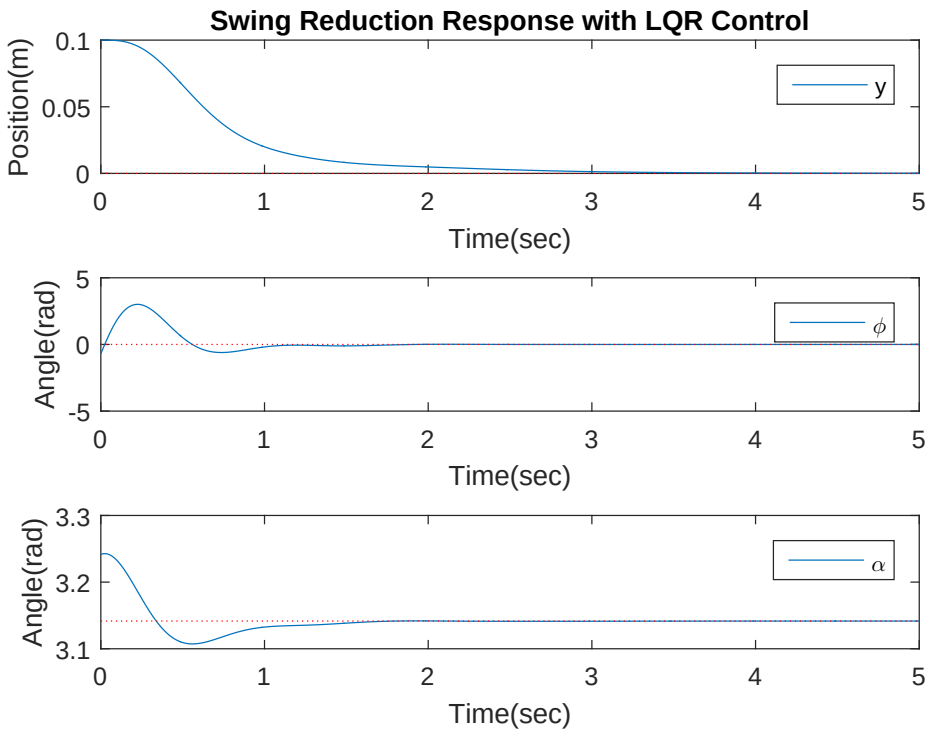


Figure 5.6: Responses of the states to reducing the swing of the pendulum. First graph shows the y position of the quadrotor, second graph shows the pitch angle of the quadrotor and the last graph shows the pendulum angle.

5.5 Gazebo

For the Gazebo simulation, the existing package `rotors_simulator` has been used to model the quadrotor to have a model with exact characteristics and inertias [43]. The pendulum, its arm and a counterweight has been added to the model using xacro programming language, which is used to build Gazebo models. The exact measurements and inertia's for the three parts have been calculated and used to create a model which is as close as possible to the real system. The controllers have been developed from zero and can be seen in Appendix B. The attitude control is represented by the Roll_pendulum, Pitch_pendulum and Yaw_pendulum notes seen

in the figure. For the position control the Altitude_pendulum node is holding the quadrotor at 1m height at any point of the simulation and the X_pendulum and Y_pendulum generate an input for the attitude controller so that the pendulum stays at the origin at any point the position control is enabled. The input goes into the Transform node, which secures that even at a heading change the desired roll and pitch angles are achieved. For the linear swing-up control, the EnergieControlLateral node generates the input for the acceleration controller which controls the roll angle of the quadrotor. The LQR_Lateral node balances the pendulum when it is at its upward vertical position. The EnergieControlLateral generates the control input heading acceleration control, which switches to the LQR_Rotational control when the pendulum angle is close to zero. The LQR_Reduce_Swing control is enabled to reduce the pendulum swing at the hovering position. All these inputs go into the Motor_Commands_pendulum node which is the decision control. It makes sure that the right topics are used to generate the rotational speeds. The flight modes can be switched with an XBox controller which is connected via the joy node. When the rotational speeds are generated, they are published to the gazebo model.

The results for the linear swing-up and balancing can be seen in Figure 5.7. The first graph shows the position results for the experiment, the second subplot shows the quadrotor angles and the last shows the pendulum angle. The solid line shows the swing-up maneuver and the dotted lines show the LQR control. At the beginning the roll angle goes from -35 to 50 degrees, which is an aggressive maneuver to get the pendulum swinging. At around 1.4 seconds the pendulum is close to the upward vertical position, where it switches to the LQR controller. During the whole process the pendulum is only swinging once, before it gets to the second swing where it is getting close enough to switch to balancing control. During the LQR control roll is much smoother, and when the pendulum is stable it slowly brings the quadrotor with the balanced pendulum to the origin. A time sequence of the experiment can be seen in Appendix C in Figure C.1. It shows a short part of the movie that has

been recorded during the simulation. This experiment has also been used for the graphs which are shown here.

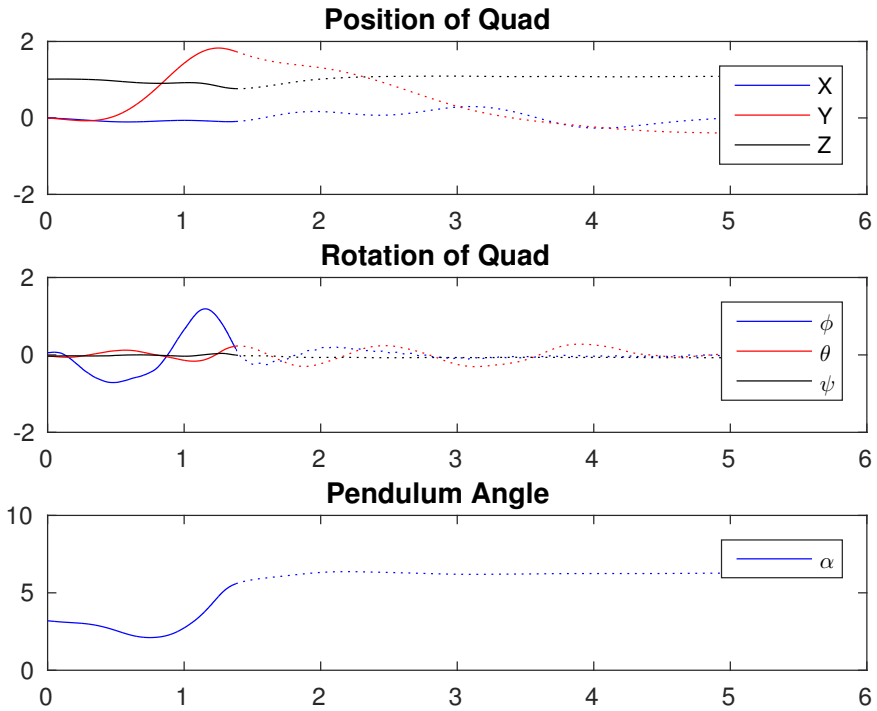


Figure 5.7: Translational position and rotational angle of quadrotor and pendulum for linear swing-up and balancing.

The results for the rotational swing-up and balancing are shown in 5.8. It should be noticed that the position during the rotation of the quadrotor is fluctuating more than during the linear control. It is getting smoother for the LQR balancing control. The quadrotor is rotating aggressively during the swing-up and continues rotating during the balancing control. This is because the heading control is the weakest control input. The quadrotor is able to balance the pendulum, however it is not able to hold its yaw at 0 degrees, which acts as a desired value for the controller. The pendulum needs a little more time and swing before the controller switches. It needs three swings, before the fourth swing brings the pendulum close enough to the

upward vertical position to switch to the LQR controller. A time sequence of the experiment can be seen in Appendix C in Figure C.2. It shows part of the movie that has been recorded during the simulation. This experiment was also used to generate the graphs.

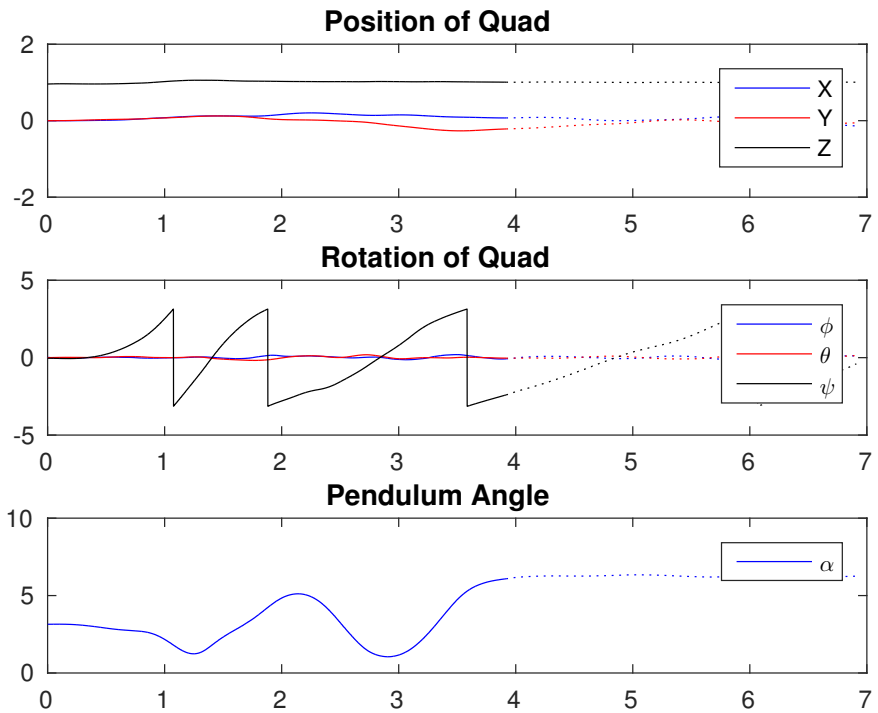


Figure 5.8: Translational position and rotational angle of quadrotor and pendulum for rotational swing-up and balancing.

In conclusion it can be said that the Gazebo simulation has shown that the system is implementable. It simulates an exact physical model of the quadrotor with the attached pendulum. All sensor noise was generated and filtered to estimate the states. The controllers have proven to be effective and robust enough to handle the aggressive maneuvers for the swing-up and balancing of the inverted pendulum. All control inputs have been saturated in the controllers, so that the model has realistic control inputs, which are actually implementable in the real world system.

5.6 Implementation

For the implementation, the two different quadrotor models have been tested with the pendulum attached to it. After testing both models, the QAV250 frame was used due to its advantage of Oneshot125 capability, payload advantage and maneuverability. Custom parts have been developed for the quadrotor to increase its visibility by Vicon and be able to attach the pendulum. The pendulum itself has been build out of carbon fiber rods, a ball bearing and custom designed and printed parts. The system with its custom parts can be seen in Figure 5.9. The pendulum has been designed lightweight by using carbon fiber. The total attachments added 39g to the 810g quadrotor, so that the total system had a weight of 849g with the battery. The length of the pendulum is 75cm. It is important to notice that the MARHES testbed area is 2.5m high, so the quadrotor has to be able to hold the altitude since there is not a lot of room in the vertical direction before the ground or ceiling is touched. It was not necessary to add a counterweight. The battery had been repositioned towards the negative on the body x-axis to create a moment around the quadrotor body origin and counteract the moment that is created by the pendulum. As a result the battery is bringing the COM back to the origin, for the case that the pendulum is not swinging. It is known that the COM is moving along the x-axis as the pendulum is rotating, however the quadrotor LL attitude control is robust enough to take this into account.

For the implementation a position control has been implemented for the takeoff, landing and hovering position hold. The linear controllers send velocities to the quadrotor. For the energy and LQR controllers the y or ψ reference will replace the position control references. However, the other states will still be controlled by the position controllers. The digital switch is controlled manually by Xbox controller which is implemented in the ROS framework and is connected to the control nodes. The rospackage used for the controller is `joy`.

Figure 5.9: Quadcopter with attached pendulum. Custom designed and printed parts to cover Odroid XU4, hold markers, attach the pendulum to quadrotor and attach pendulum to ball bearing.

For linear movement the energy control and LQR control are giving the acceleration in y, and for the rotational movement it is giving the heading acceleration. It is assumed in the model that these states can be decoupled. For this reason, a position control for y and ψ has been designed. First the pendulum with a linear movement has been tested and the result can be seen in Figure 5.10. The reference value is designed by the following control

$$\mathrm{y}_{\mathcal{A}} = \sin(kt_d), \tag{5.38}$$

where t_d is the time duration. The duration starts counting when the control is activated. The results look promising due to the fact that the quadrotor is following

the reference position. More importantly it can be noticed that x, z and ψ are stable and stay close to their references. The pendulum is moving as a result of the linear motion. However, since the controller does not try to get it to the upward vertical position it stays around its downward vertical equilibrium point. This result looks promising and gives confidence to take the next step.

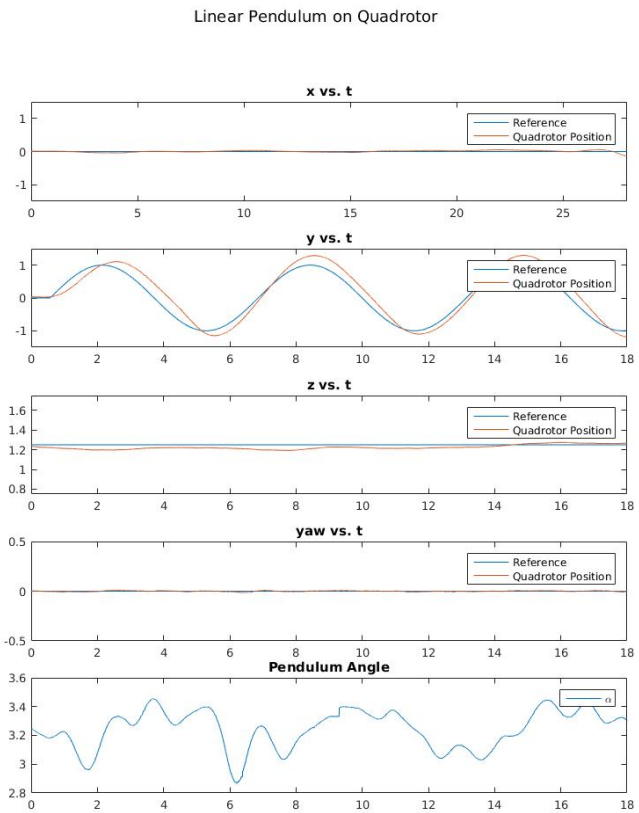


Figure 5.10: Linear position control of quadrotor with pendulum attached.

After testing the linear movement with the quadrotor attached, the rotational movement has been tested. Similarly to the linear position generation a position has been generated for the ψ position. The position has been generated using

$$\psi = 1.5 * \sin(kt_d), \tag{5.39}$$

.

The position is amplified by 1.5 due to experimenting and the engineering judgment achieved by the tests. Figure 5.11 shows the results of the test. It can be seen that x and y are moving when the quadrotor is rotating its heading. The quadrotor is able to hover around the origin, however its reaction could be improved. The altitude is stable around 1m. The pendulum angle is moving due to the rotation.

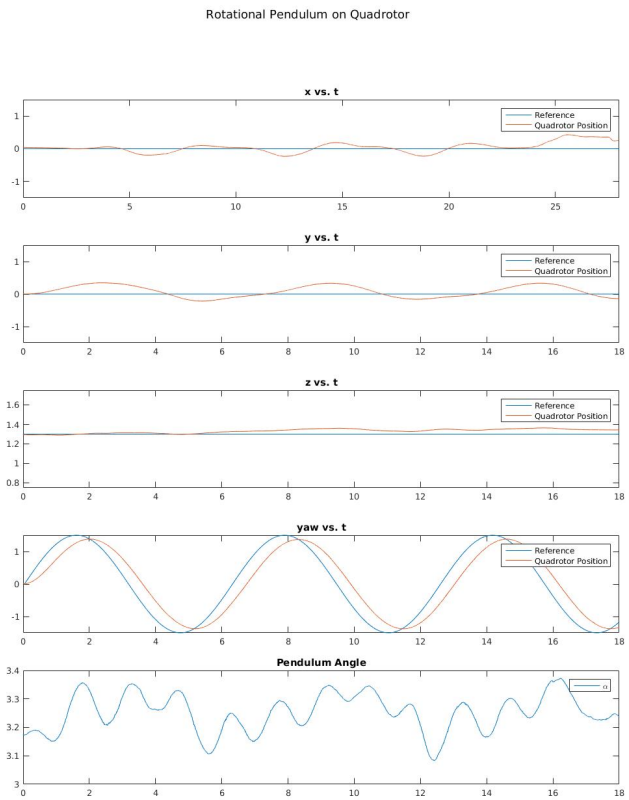


Figure 5.11: Rotational position control of quadrotor with pendulum attached.

The results of the linear and rotational position control give promising results.

It has been shown that the states can be decoupled. Improvements could be made in the x and y position when the quadrotor is rotating. It can be seen that the quadrotor has inputs to the pendulum, however the pendulum is light enough to not affect the quadrotor. The LL and HL controllers are robust enough to counteract the change in COM during the movement.

Chapter 6

Conclusions, Contributions and Improvements

6.1 Conclusions

In this thesis a ROS based framework for quadrotors at the MARHES lab has been developed. The framework makes the UAV interface user friendly and existing attitude and position controllers can be used by the users. Additionally ROS makes it possible to control multiple UAVs at once. This framework is state of the art for robotics laboratories and helps to use the aerial vehicles for different applications.

The different hardware and software used for this thesis has been explained in detail. The quadrotor model has been derived using Newton-Euler approach with rigid body assumptions. The system architecture shows the implementation of the framework on two quadrotors with different FC. It demonstrates that even though the quadrotors have different LL controllers, they can be controlled similarly on the HL side, which simplifies the implementation of them in a network.

Two applications are explained that were used to demonstrate the flexibility of the quadrotor. The MAST and ASAP project make use of the new framework. For the MAST project a linear position controller has been developed and implemented. Different velocity estimators have been tested and implemented, the best performing estimator is used for the position control. A demo has been designed and explained where the quadrotor hops between different ground vehicles to extract data from them. For the ASAP project a trajectory generation and implementation algorithm has been tested and documented. The implementation architecture shows the use of Matlab in the framework. Tests for an open loop reachability analysis are discussed.

A framework for a pendulum on a quadrotor is presented. It explains the possibility of swinging-up and balancing a rotational and linear pendulum with a quadrotor as the actuator. The complete model is derived using Lagrangian. Control algorithms for the hybrid controller have been developed and tested for stability. First Matlab is used to calculate the LQR gain matrices which are used in the Gazebo simulation to demonstrate the system. First steps and tests for a real system implementation are documented.

6.2 Contributions

The main contributions of this thesis are

- ROS UAV Framework: This thesis developed a framework for on-board microprocessors on quadrotors which use Robot Operating System. They get position data from Vicon and it is possible to control them with standard ROS messages. Additionally it is possible to communicate with them on a ground station using ROS and/or Matlab's Robotic Toolbox.
- Architecture: The system is explaining the implementation architecture in de-

tail. It can be seen that the HL side is similar, even though the LL control side uses different FC.

- Position Control: A linear position control is implemented. It shows good results, whereas the focus was on stable flying around the desired position for the MAST project. This position control will benefit the implementation of optical communication devices on the quadrotor.

- Trajectory Tracking: Trajectory tracking in the framework of the ASAP project is implemented. It generates smooth flight paths using 3rd order polynomials in Matlab. The Matlab Robotics Toolbox has been used to send the trajectory coefficients to the quadrotor.

- Pendulum on a Quadrotor: A framework for an off-centered pendulum on a quadrotor has been developed. The model has been derived and a promising control strategy developed. The simulations show promising results and the implementation has been discussed.

6.3 Improvements

The quadrotors are limited by the Vicon area that is provided. The framework has the possibility to switch Vicon position feedback with GPS feedback, so that it is possible to go outdoors. This would also make it possible to add more agents to the network of UAV's.

The LL controllers present some limitations by themselves. It is possible to control both quadrotors that were developed in the framework via direct motor control. This could improve the responsiveness and overall output of the system. Much research has been done in ESC protocols. Being able to switch to faster protocols (Oneshot42, Multishot) or digital protocols (DSHOT150, DSHOT300, DSHOT600,

DSHOT1200) could improve performance as well as reduce noise between the LL controller and the ESC.

Different controllers could be implemented in the framework. The control inputs to the framework could be changed and probably make the quadrotor faster, more agile and more stable.

Cooperative applications and algorithms could be implemented to test the network ability of the framework.

Appendices

Appendix A

Graphs for Trajectory Tracking a Figure Eight

This results of flying a figure eight trajectory have been documented here. The quadrotor has been tested at different speeds. The different gains, k, control the speed of the vehicle. The UAV has been tested for the gains 0.5,1.0,1.5 and 2. The paragraphs show each position of the quadcopter at specific times and how it performs against its reference.

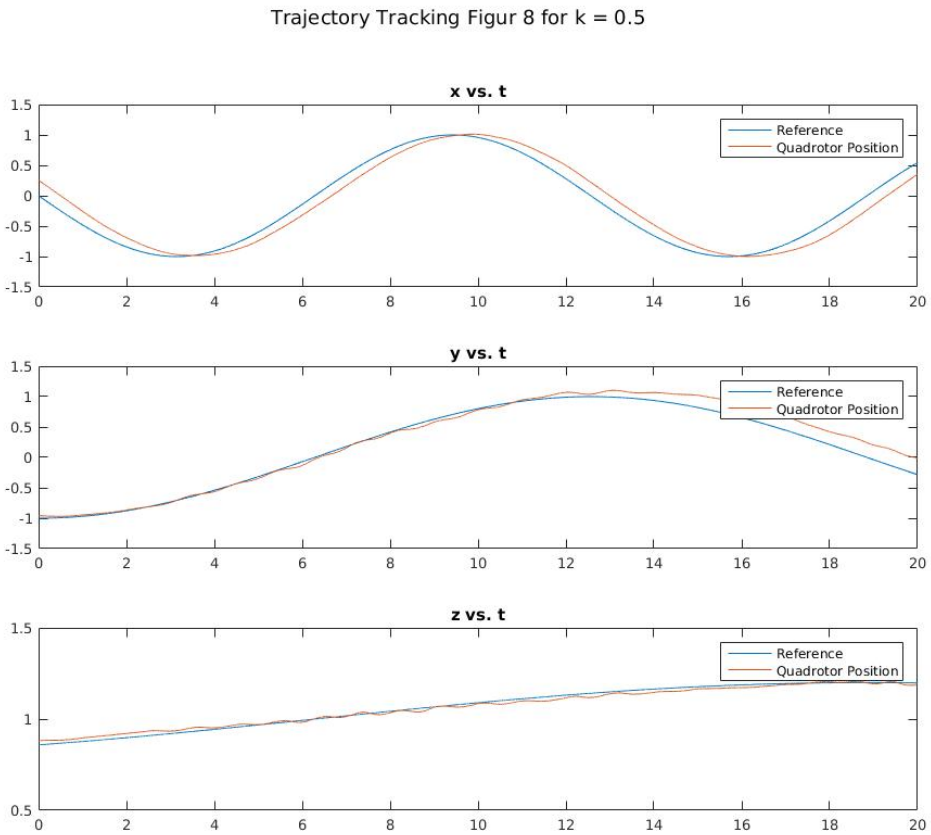


Figure A.1: Position vs. time graphs for trajectory tracking figure eight with gain k=0.5

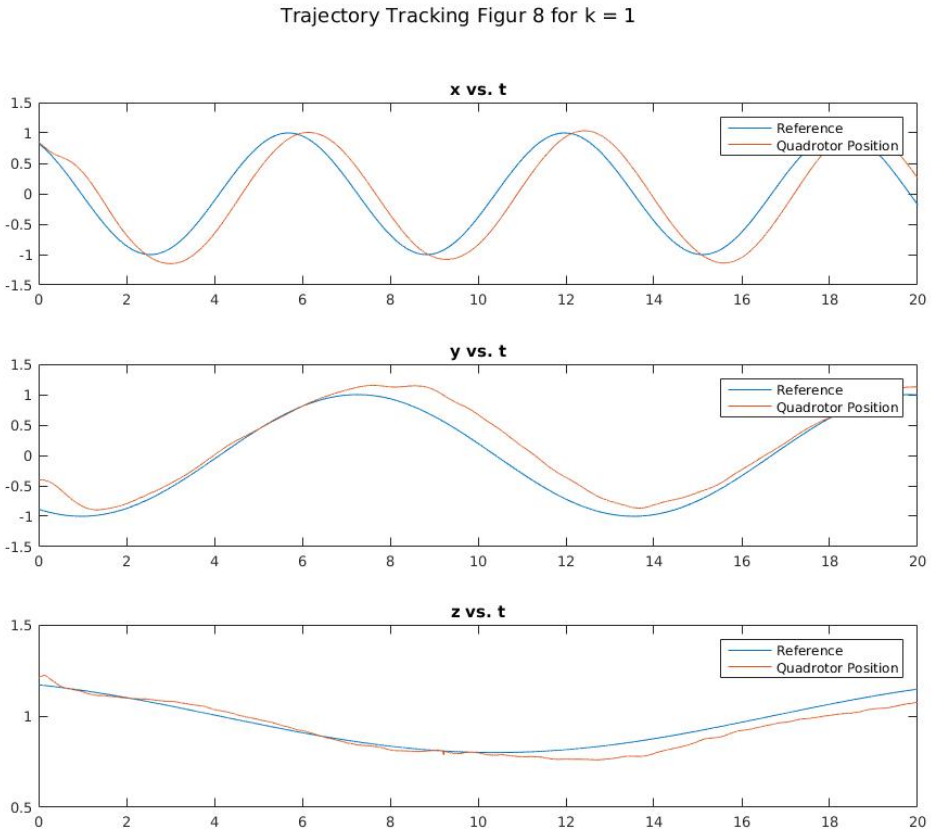


Figure A.2: Position vs. time graphs for trajectory tracking figure eight with gain k=1.0

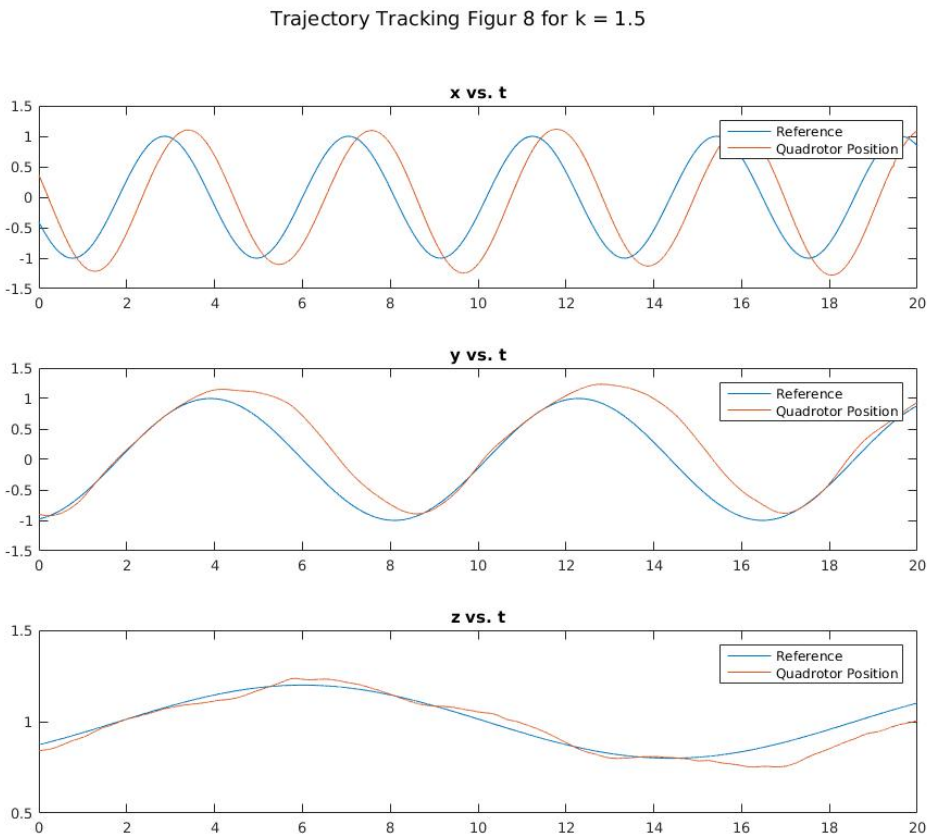


Figure A.3: Position vs. time graphs for trajectory tracking figure eight with gain k=1.5

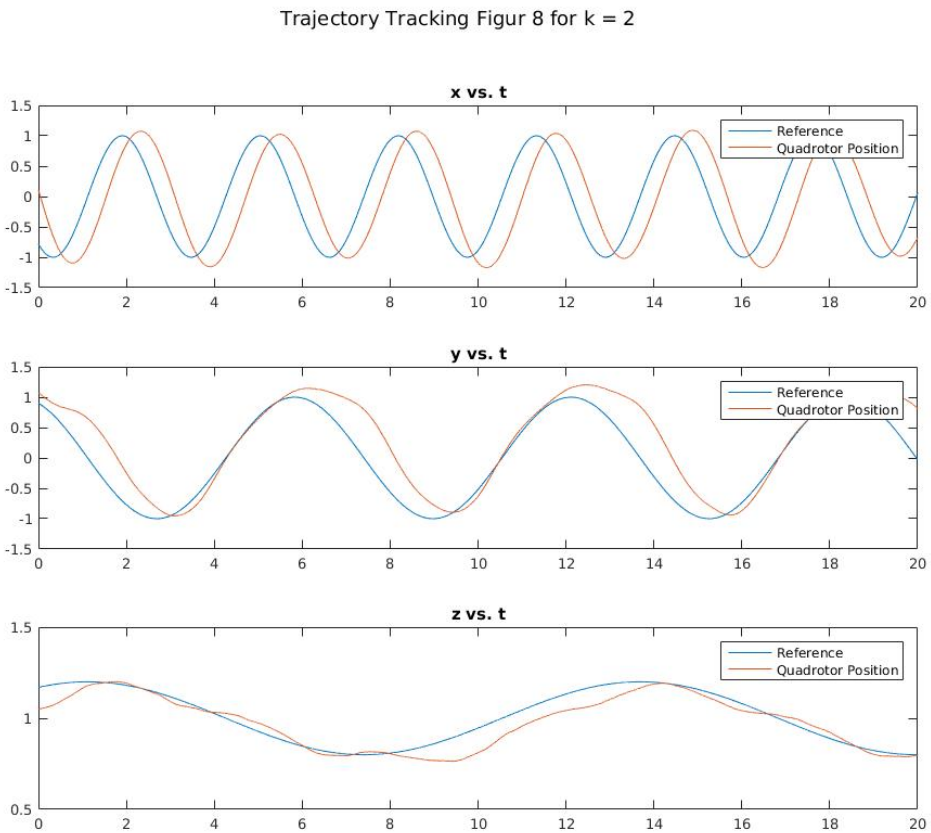


Figure A.4: Position vs. time graphs for trajectory tracking figure eight with gain k=2.0

Appendix B

Gazebo Simulation

The Gazebo simulation has been developed with the help of the rotors_gazebo package developed by students from ETH-Zurich. This has the advantage that the hummingbird quadrotors have exact characteristics, so that the simulation will be as close as possible to the real world. The pendulum has been added to the quadrotor with exact measurements and inertia's. The controllers for every level have been developed. The control tree can be seen in Figure B.1.

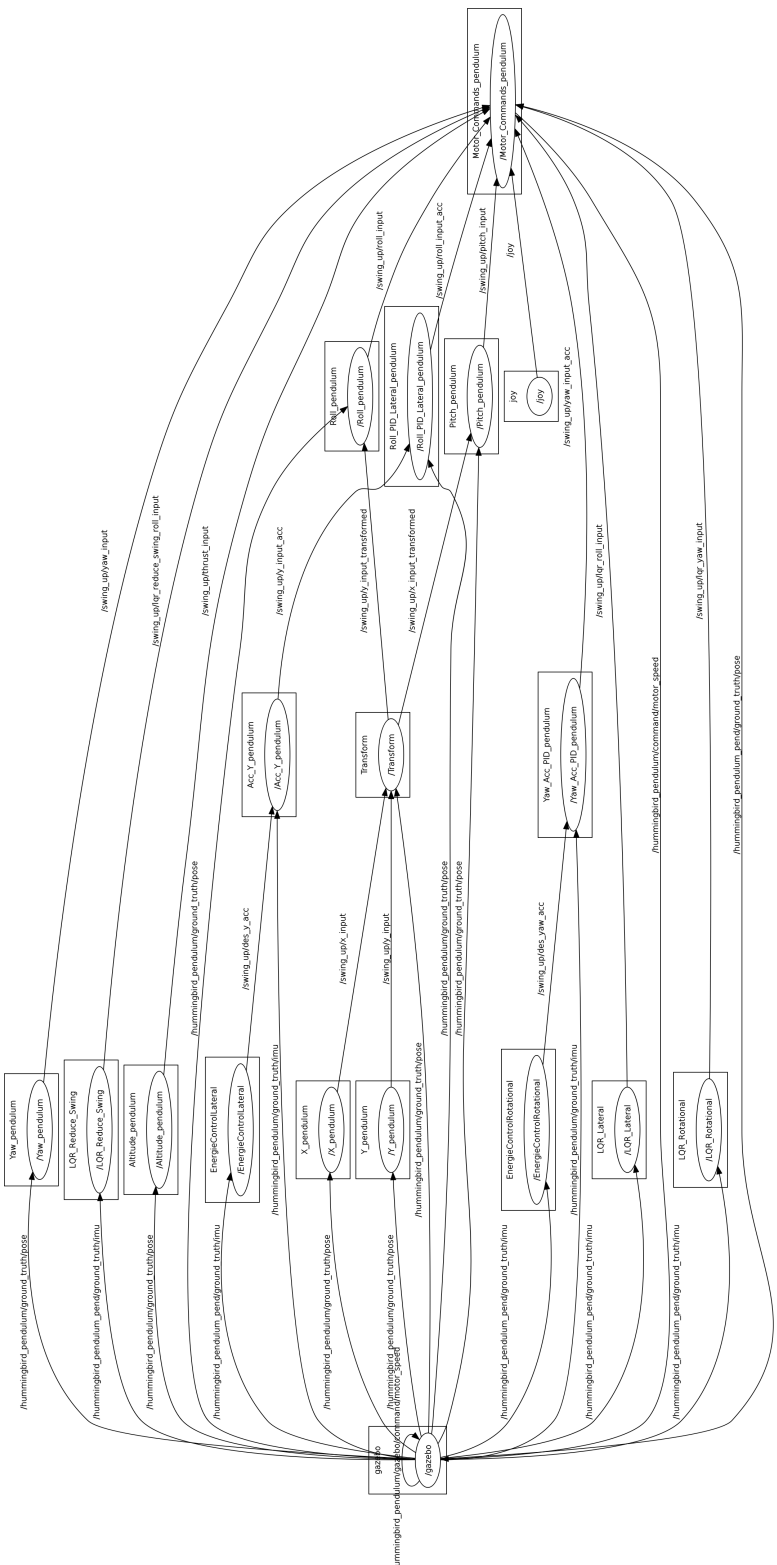


Figure B.1: Control tree of Gazebo simulation with all control nodes and used topics.

Appendix C

Picture Sequence of Gazebo Simulation

The picture sequences taken to display the swing-up and balancing control of the pendulum has been developed using Gazebo and Matlab. First a video of the complete simulation has been recorded. The video was cut into the rotational and linear simulation. The video was imported into Matlab and separated into pictures. The pictures have been converted into a sequence to show the process. The linear swing-up and balancing can be seen in Figure C.1 and the rotational can be seen in C.2.

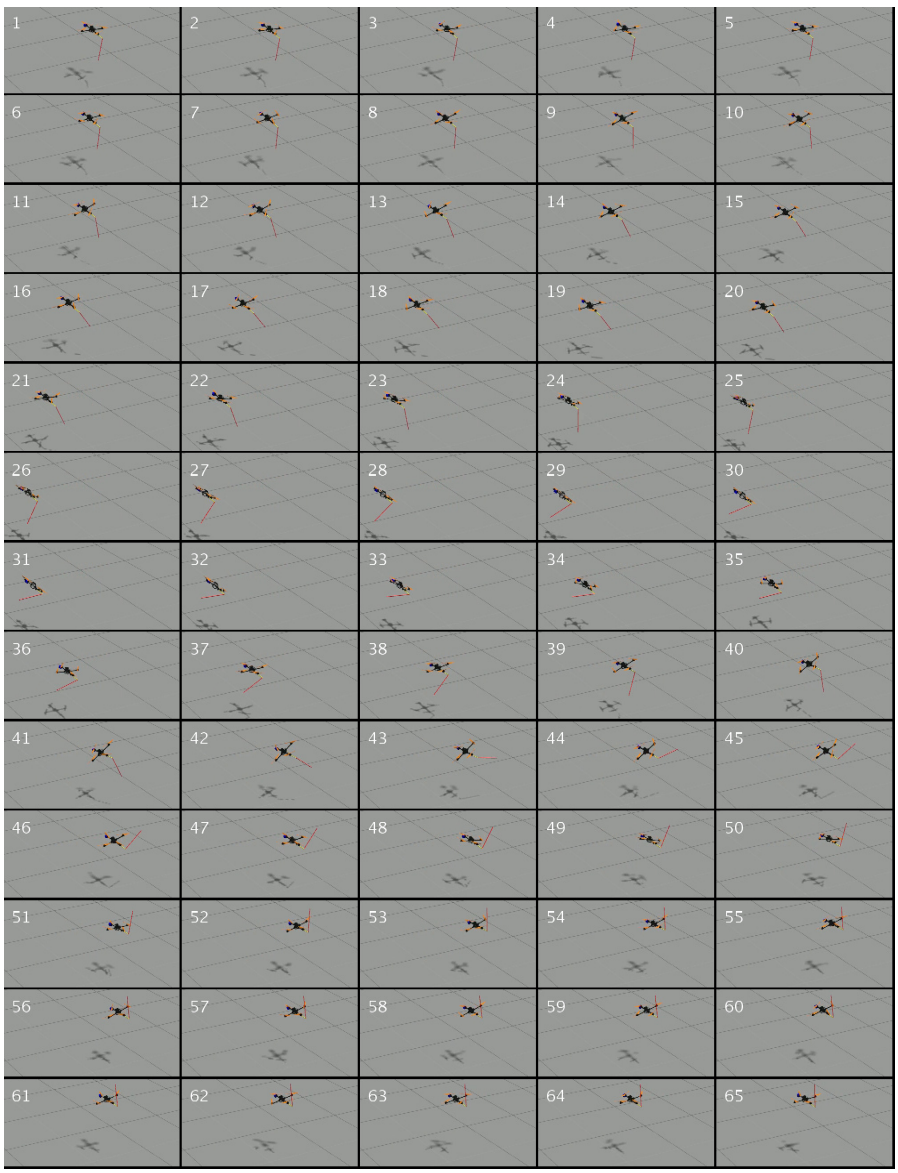


Figure C.1: Picture sequence of swing-up and balancing simulation the linear pendulum on a quadrotor in Gazebo.

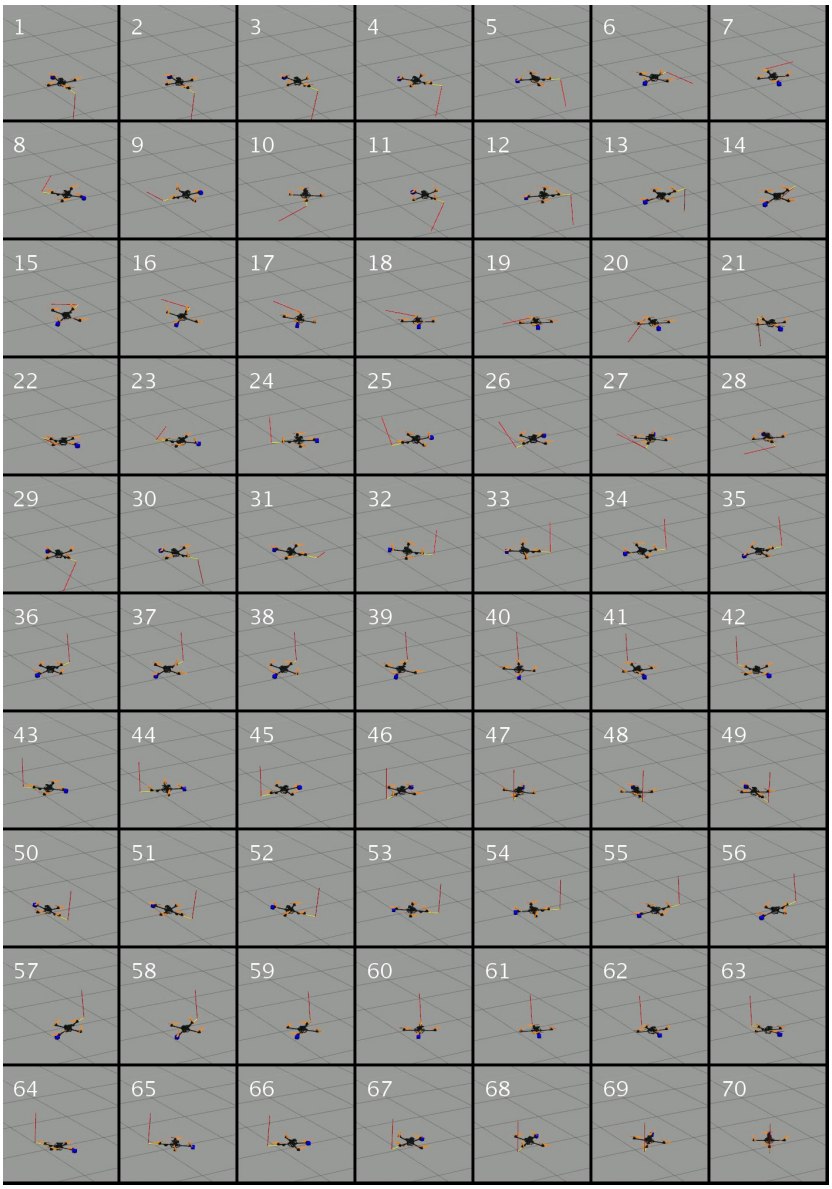


Figure C.2: Picture sequence of swing-up and balancing simulation the rotational pendulum on a quadrotor in Gazebo.

References

[1] "Documentation." `https://www.mathworks.com/help/robotics/examples/exchange-data-with-ros-publishers-and-subscribers.html`.

[2] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin, "The stanford testbed of autonomous rotorcraft for multi agent control (starmac)," in *Digital Avionics Systems Conference, 2004. DASC 04. The 23rd*, vol. 2, pp. 12–E, IEEE, 2004.

[3] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics & Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[4] M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J. P. How, and J. Vian, "The mit indoor multi-vehicle flight testbed," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 2758–2759, IEEE, 2007.

[5] S. Lupashin, A. Schöllig, M. Hehn, and R. D'Andrea, "The flying machine arena as of 2010," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 2970–2971, IEEE, 2011.

[6] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard, *Modeling the Quad-Rotor Mini-Rotorcraft*, pp. 23–34. London: Springer London, 2013.

[7] S. Bouabdallah, A. Noth, and R. Siegwart, "Pid vs lq control techniques applied to an indoor micro quadrotor," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, pp. 2451–2456, IEEE, 2004.

[8] T. Bresciani, "Modelling, identification and control of a quadrotor helicopter," *MSc Theses*, 2008.

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

[10] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *The International Journal of Robotics Research*, vol. 31, no. 5, pp. 664–674, 2012.

[11] P. J. Cruz Davalos, *Real-time control architecture for a multi UAV test bed*. PhD thesis, 2013.

[12] W. Neeley, *Design and Development of a High-Performance Quadrotor Control Architecture Based on Feedback Linearization*. PhD thesis, 2016.

[13] J. Boren and S. Cousins, "Exponential growth of ros [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 1, pp. 19–20, 2011.

[14] S. Cousins, B. Gerkey, K. Conley, and W. Garage, "Sharing software with ros [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 12–14, 2010.

[15] S. Edwards and C. Lewis, "Ros-industrial: applying the robot operating system (ros) to industrial applications," in *IEEE Int. Conference on Robotics and Automation, ECHORD Workshop*, 2012.

[16] A. A. El-Badawy and M. A. Bakr, "Quadcopter aggressive maneuvers along singular configurations: an energy-quaternion based approach," *Journal of Control Science and Engineering*, vol. 2016, p. 4, 2016.

[17] M. Achtelik, T. Bierling, J. Wang, L. Hocht, and F. Holzapfel, "Adaptive control of a quadcopter in the presence of large/complete parameter uncertainties," *Infotech@ Aerospace*, pp. 29–31, 2011.

[18] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation: Safe and efficient load manipulation with aerial robots," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 69–79, 2012.

[19] P. J. Cruz and R. Fierro, "Towards optical wireless communications between micro unmanned aerial and ground systems," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pp. 669–676, IEEE, 2015.

[20] B. HomChaudhuri, A. P. Vinod, and M. M. Oishi, "Computation of forward stochastic reach sets: Application to stochastic, dynamic obstacle avoidance," in *American Control Conference (ACC), 2017*, pp. 4404–4411, IEEE, 2017.

[21] A. P. Vinod, B. Homchaudhuri, and M. M. Oishi, "Forward stochastic reachability analysis for uncontrolled linear systems using fourier transforms," in *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, pp. 35–44, ACM, 2017.

[22] J. Aracil, J. Acosta, and F. Gordillo, "A nonlinear hybrid controller for swinging-up and stabilizing the furuta pendulum," *Control Engineering Practice*, vol. 21, no. 8, pp. 989–993, 2013.

[23] K. J. Åström and K. Furuta, "Swinging up a pendulum by energy control," *Automatica*, vol. 36, no. 2, pp. 287–295, 2000.

[24] V. Sukontanakarn and M. Parnichkun, "Real-time optimal control for rotary inverted pendulum," *American journal of applied sciences*, vol. 6, no. 6, p. 1106, 2009.

[25] S. Awtar, N. King, T. Allen, I. Bang, M. Hagan, D. Skidmore, and K. Craig, "Inverted pendulum systems: rotary and arm-driven-a mechatronic system design case study," *Mechatronics*, vol. 12, no. 2, pp. 357–370, 2002.

[26] M. Hehn and R. D'Andrea, "A flying inverted pendulum," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 763–770, IEEE, 2011.

[27] D. Brescianini, M. Hehn, and R. D'Andrea, "Quadrocopter pole acrobatics," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3472–3479, IEEE, 2013.

[28] "Turtlebot2." `http://www.turtlebot.com/turtlebot2/`.

[29] "Pioneer 3-at." `http://www.mobilerobots.com/ResearchRobots/P3AT.aspx`.

[30] "4x4 monster pick-up txt-1." `http://www.tamiya.com/english/products/58280txt_1/txt_1.htm`.

[31] "Posts about octoroach on robotics." `https://walterfarah.wordpress.com/tag/octoroach/`.

[32] "miniroach." `http://marhes.unm.edu/?page_id=1206`.

[33] "Asctec hummingbird /// research and development uas for swarming and control theory.." `http://www.asctec.de/en/uav-uas-drones-rpas-roav/asctec-hummingbird/`.

[34] "Baxter collaborative robots for industrial automation." http://www.rethinkrobotics.com/baxter/.

[35] Vicon, "Motion capture systems." https://www.vicon.com/.

[36] "Odroid — hardkernel." http://www.hardkernel.com/main/products/prdt_info.php.

[37] "Jetson tk1 embedded developer kit from nvidia." http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html.

[38] A. Industries, "Raspberry pi zero w." https://www.adafruit.com/product/3400.

[39] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, 2012.

[40] "Pixracer autopilot." https://pixhawk.org/modules/pixracer.

[41] "Wiki." http://wiki.ros.org/ROS/Introduction.

[42] Osrf, "Why gazebo?." http://gazebosim.org/.

[43] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. Cham: Springer International Publishing, 2016.

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

[45] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles," *IEEE Robotics and Automation magazine*, vol. 20, no. 32, 2012.

[46] M. K. Habib, W. G. A. Abdelaal, M. S. Saad, *et al.*, "Dynamic modeling and control of a quadrotor using linear and nonlinear approaches," 2014.

[47] J. Li and Y. Li, "Dynamic analysis and pid control for a quadrotor," in *Mechatronics and Automation (ICMA), 2011 International Conference on*, pp. 573–578, IEEE, 2011.

[48] M. A. Al-Alaoui, "Al-alaoui operator and the new transformation polynomials for discretization of analogue systems," *Electrical Engineering*, vol. 90, no. 6, pp. 455–467, 2008.

[49] R. P. K. Jain, "Transportation of cable suspended load using unmanned aerial vehicles: A real-time model predictive control approach," 2015.

[50] P. Kotaru, G. Wu, and K. Sreenath, "Dynamics and control of a quadrotor with a payload suspended through an elastic cable," in *American Control Conference (ACC)*, to appear, 2017.

[51] K. J. Åström, "Hybrid control of inverted pendulums," in *Learning, control and hybrid systems*, pp. 150–163, Springer, 1999.

[52] R. Fierro, F. L. Lewis, and A. Lowe, "Hybrid control for a class of underactuated mechanical systems," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 29, no. 6, pp. 649–654, 1999.