# Aerial Manipulation for Pick-And-Place Applications

Modelling, Estimation, and Control of Nonlinear Systems

Rasmus Nymark Skjelsager

MSc. Thesis - Control and Automation

June 2, 2023

Aalborg University
Electronics and IT

AALBORG UNIVERSITY
STUDENT REPORT

**MSc. Thesis**
**Electronics and IT**
Aalborg University
https://www.aau.dk

**Title:**
Aerial Manipulation for Pick-and-Place Applications

**Theme:**
Modelling, Estimation, and Control of Nonlinear Systems

**Project Period:**
Spring 2023

**Project Group:**
Group 1039

**Participant(s):**
Rasmus Nymark Skjelsager

**Supervisor(s):**
Henrik Schiøler

**Copies:** 1

**Page Numbers:** 81

**Date of Completion:**
June 2, 2023

**Abstract:**

This work treats the design, modelling, and state estimation for an aerial manipulator. Initially, mechanical and electrical designs are made for an aerial manipulator and a payload. This is followed by modelling of both the system dynamics and the sensors. The modelling includes both descriptions of Euler-Lagrange and Newton-Euler dynamics. Several physical constants are estimated, with the goal of using them for control at a later stage. A system for taking 6-DOF camera measurements is implemented and used for a state estimator based on the UKF. Due to time constraints no controller was implemented. Testing showed that the Euler-Lagrange dynamic model worked well, and that the camera measurement system records corrupted data, breaking the state estimator.

# Contents

# 1 | Introduction

The use of Unmanned Aerial Vehicles (UAVs) for inspection, data collection, and surveying tasks has been rapidly increasing and the applications are numerous. The potential for UAVs can be extended to new fields if made able to complete tasks requiring interaction with the environment, such as lifting, pick-and-place, assembly, cleaning, or collecting samples [1].
Specific examples are transportation of goods, maintenance of tall structures such as powerlines or tall buildings [2], assembly or installation of equipment (sensors) at altitude, and obtaining samples in inaccesible or hazardous locations [3]. This also includes industrial lifting operations for wind turbine maintenance which is the focus of, for example, Danish aerospace company Airflight ApS [4].

Adding and controlling a manipulator attached to a UAV - an Unmanned Aerial Manipulator (UAM) - has proven more difficult due to the lack of a stable base, which complicates the control task [1]. Other problems include locating objects and pose-estimating objects in relation to the UAM. Current approaches include using visual markers on objects [5] and visual servoing for gripping [6], [7].
The manipulator and its gripper are often custom designed parts, which is due to the special requirements of a UAM, such as low mass. Several different types of grippers exist and have been tested, and the type of gripper required depends on the specific objects to be gripped in a given application [1], [8].

The various tasks requires different levels of coupling between the UAM and the manipulated object. In [9] these are categorized as:

- Momentary coupling: tasks with objects that can be picked up and manipulated while airborne. Typically transportation, lifting or pick-and-place tasks.
- Loose coupling: tasks where the UAM manipulates a static object attached to the environment while itself remaining airborne. Typically assembly, insertion, pushing or pulling tasks.
- Strong coupling: tasks where the UAM lands and perches on an object while interacting with it.

Control algorithms for position control and trajectory tracking have been demonstrated for both manipulators with few DoFs (1DoF, 2DoF), and manipulators with more (3+ DoF). Applied control techniques in existing literature include: PID [10], LQR [11], [12], ADRC with backstepping [13], adaptive control [10], sliding mode,

and MPC [3], [14]. More complex controllers tend to provide better performance but computational time has been an issue [3], [14]. This is a typical issue and also extends to state estimators for the UAM problem, where for example Unscented Kalman Filters (UKF) have been shown to perform well [12]. However, they come at the cost of increased computation time compared to Extended Kalman Filters (EKF).

The control of aerial manipulators is a promising field of research with an abundance of possible applications. Development of accurate control strategies that are able to handle the complex dynamics and complete the manipulation tasks is required for real-world implementations of aerial manipulation. In order to limit the scope of tasks this work will focus on generic pick-and-place tasks that require picking up and moving an object to a different position. This leads to the problem statement:

> *How can an aerial manipulator be designed, modelled and controlled to be able to complete pick-and-place tasks?*

## 1.1 Project objectives

The objectives of this project include:

- Design and implementation of a prototype aerial manipulator for laboratory testing.
- Mathematical modelling of the system.
- Design and implementation of a state estimator for the UAM.
- Design and implementation of a control strategy for the UAM.
- Demonstration and validation of the above points through simulation and experiments.

These points will be covered in the following chapters.

# Part I

# System & Model

# 2 | System

This chapter will provide descriptions of the components used to build the prototype system. This includes descriptions of both mechanical and electrical components, including sensors and actuators, and how these interact. The goal is to provide an overview of the system. Overall, the UAM is composed of a quadrotor, acting as the base of the robot, and a manipulator attached to the quadrotor. This is illustrated in Fig. 2.1. The quadrotor and the manipulator will be described separately in the following sections.

**Figure 2.1:** 3D render of the UAM with a two-link manipulator with a simple hook as end-effector.

| Name | Description | Amount | Mass [g] |
|---|---|---|---|
| Base (bottom) | Bottom plate of the base with power distribution | 1 | 54 |
| Base (top) | Top plate of the base, carries electronics | 1 | 22 |
| Motor arm | Arm connecting BLDC motors with the base | 4 | 44 |
| Legs | Legs for the UAM to stand | 4 | 11 |
| Propeller | T-motor T1045 propeller 10×4.5" | 4 | 12 |
| Servo cage | Cage for attaching a servo to the base | 1 | 26 |
| TPU Standoffs | Standoffs for mounting electronics | 4 | 1 |
| Propeller guards | For shielding propellers in case of a crash | 4 | 11 |

**Table 2.1:** Overview and description of mechanical components of the quadrotor, including amount of the component used in the assembly along with the mass of each individual component.

## 2.1 Quadrotor

The quadrotor is the core part of the UAM, providing the base link to which all components of the UAM is attached, including motors, the battery, avionics/flight controller, and the manipulator itself.

### 2.1.1 Mechanical Parts

The quadrotor is built up of multiple parts. In Table 2.1 an overview of the components is displayed. Certain small components such as cables, connectors, screws, bolts, and nuts are excluded from this table. The two base plates are effectively reinforced FR4 PCB plates. The top plate has no electrical connections whereas the bottom plate has built-in power distribution traces for connecting the Electronic Speed Controllers (ESCs) and the battery.

The arms carrying the motors were designed in SolidWorks and the design was optimized using the Finite Element Analysis (FEA) and topology optimization features in SolidWorks to reduce the weight. This is illustrated in Fig. 2.2.

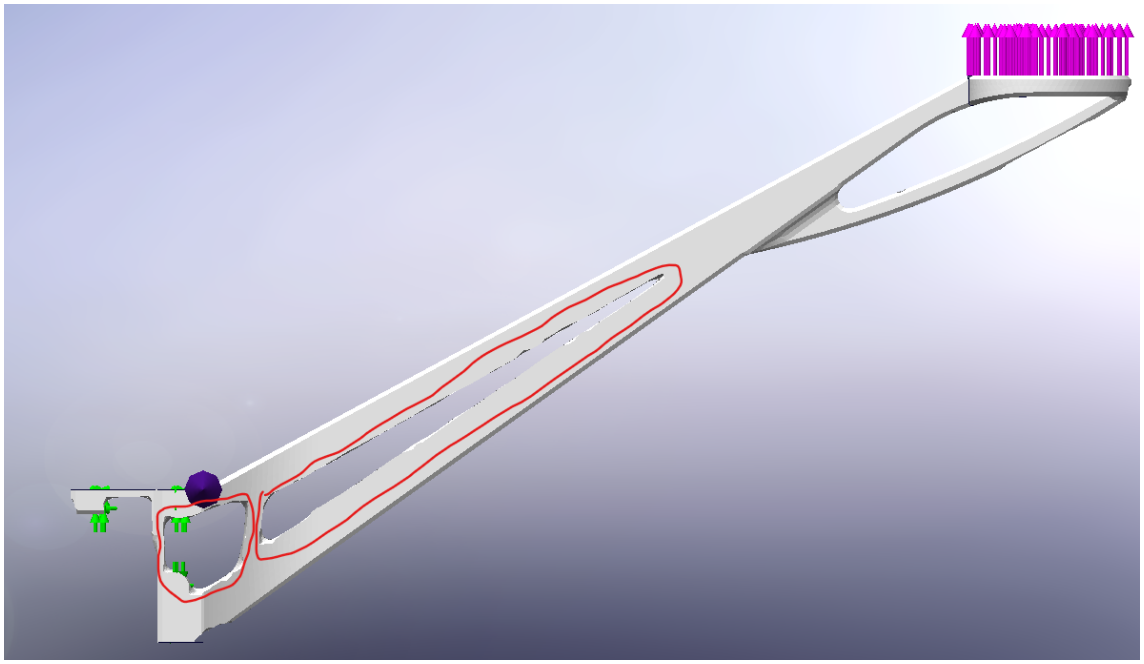


**Figure 2.2:** A screenshot from the SolidWorks Topology Optimization tool. The areas encircled in red are the areas marked by SolidWorks as eligible for removal. Before running the analysis, these areas were solid.

The FEA and topology optimization was carried out by applying the motor and propeller manufacturer's indicated maximum thrust of the combination of motor and propeller. This is considered the worst case scenario of applied forces under normal operating conditions. Other parts were simply designed with low weight in mind, but as they do not experience the same amount of applied forces, FEA was not applied. The rotor arm is designed to raise the rotor a small distance above the base. This slightly decreases the lever arm effect of the rotors on the body, effectively reducing roll-, and pitch-controllability. This is expected to result in dynamics that are somewhat dampened, since actuators have to use more control effort to rotate the quadrotor.

The stand-offs for the electronics were 3D-printed in a flexible plastic (TPU with a Shore-hardness of 98A) and are designed as two cones stacked on each other in an "hourglass" fashion to be able to absorb vibrations from the motor.

### 2.1.2 Electronics & Sensors

The electronics included on the quadrotor are listed in Table 2.2.

| Name | Description | Amount | Mass [g] |
|---|---|---|---|
| 4S LiPo Battery 5000mAh | LiPo battery pack | 1 | 436 |
| T-Motor Air 2216 KV880 | Brushless DC motors | 4 | 68 |
| T-Motor Air 20A ESC | Electronic Speed Controllers for the motors | 4 | 19 |
| Teensy 4.0 | Microcontroller acting as flight controller | 1 | 6 |
| LSM6DSOX+LIS3MDL IMU | High-precision combined accelerometer, gyroscope, and magnetometer | 1 | 4 |
| GY87 IMU | Combined accelerometer, gyroscope, and magnetometer | 1 | 1 |
| BMP388 Barometer | High-precision barometer for altitude measurements | 1 | 2 |
| SP3485 (RS485) Transceiver | Transceiver for communication with manipulator servos | 1 | 1 |
| APC220 433MHz Transceiver | Wireless communication module for general purpose user IO | 1 | 11 |
| LM2596 Buck Converter | Voltage regulator for supplying other electronics | 2 | 16 |
| Raspberry Pi 4 Model B | Onboard computer for image processing | 1 | 49 |
| 16MP IMX519 PDAF & CDAF Autofocus Camera | Small camera that can be connected to the Raspberry Pi | 1 | 7 |
| Generic LiPo Battery Alarm | Battery level indicator | 1 | 7 |
| Custom PCB | Custom made PCB for connecting the various electronic components | 1 | 36 |

**Table 2.2:** Overview and short description of electronic components of the quadrotor

The flight controller is based on a Teensy 4.0 microcontroller (MCU), due to its small size and high performance, as well as its support for the Arduino IDE and libraries, making development easy. The sensor suite consists of two different Inertial Measurement Units (IMUs), and a high-precision barometer acting as an altimeter. To allow the system to communicate with external devices a wireless 433MHz transceiver module is included on-board and connected to the MCU. Furthermore, an RS485 module is included since the manipulator servos communicate through this protocol. These are all connected on a custom PCB illustrated in Fig. 2.3.

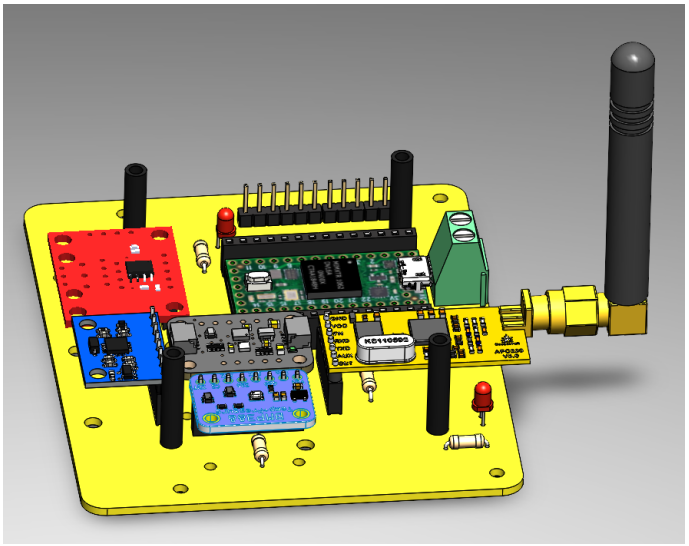


**Figure 2.3:** A 3D-render of the electronic components installed on a custom PCB to connect them all.

Each of the transceiver modules requires a UART channel, whereas the sensors are all connected on the same I2C bus. Power is supplied through the screw terminal from an LM2596 regulator module. The ESCs also connect to this board as they require control signals from the flight controller. The electrical interconnections of the system may be found in Appendix A. A Raspberry Pi 4 Model B (RPI) will be used for collecting data from the MCU and for image processing which will be used for positioning. The RPI is to be mounted atop the four tall black standoffs displayed in Fig. 2.3.
3D models exist of all the electronics. These 3D models will be used when approximating the inertia matrix of the system at a later stage. All the components in this section are assembled into a quadrotor as displayed in Fig. 2.4.

**Figure 2.4:** A 3D-render of the quadrotor with electronics (including battery, motors, and ESCs) installed.

## 2.2 Manipulator

The manipulator consists of several mechanical parts and two servos. An overview of the components is presented in Table 2.3. It is made with 2 degrees of freedom (DoF), but it is possible to treat it as either a 1 DoF or 0 DoF (static object) as well. The first joint of the manipulator rotates around yaw-axis of the quadrotor. This is an uncommon configuration for aerial manipulators. The reason for choosing such a design is that it allows the system to use the manipulator to exert torques about its yaw-axis. Yaw motion for a quadrotor typically has significantly weaker controllability compared to pitch and roll, since the rotor torques are mainly due to the drag on the propellers. The second joint axis is rotated 90 degrees compared to the first joint axis. In principle this joint can also be used to control the quadrotor attitude, however the mass and inertia of the second link is significantly lower. For that reason this is only really an option if the UAM is carrying a payload. The end-effector of the manipulator is a simple hook, which was chosen due to its simplicity and low weight. A downside to the hook is that it does not fix the payload in place like a typical robotic gripper would, but rather allows it to swing like a pendulum. The forces act as a disturbance on the quadrotor, and without sufficient actuation to offset them, they may cause a crash. This is mainly a problem with heavy payloads. As a proof of concept however, this work will not deal with particularly heavy payloads, thus largely eliminating this issue. However, it should be considered in industrial applications.

| Name | Description | Amount | Mass [g] |
|---|---|---|---|
| Dynamixel XH430-W210-R | Smart servo with RS485 communication | 2 | 84 |
| Link 1 Tube | Carbon fiber tube linking joints 1 and 2 | 1 | 4 |
| Link 2 Tube | Carbon fiber tube linking joint 2 with the end-effector | 1 | 8 |
| Hook | The end-effector | 1 | 4 |
| Servo-link 1 connector | Connector between joint 1 servo and link 1 tube | 1 | 6 |
| Servo-joint 2 cage | Cage attaching to link 1, holding the servo for joint 2 | 1 | 22 |
| Servo-link 2 connector | Connector between joint 2 servo and link 2 tube | 1 | 7 |
| M3x30 Bolts | For holding the links in place | 8 | 0.5 |
| M2x5 Bolts | For attaching link connectors to the servo horns | 16 | <0.1 |
| M2.5x8 Bolts | For keeping the servos in place | 16 | 0.1 |

**Table 2.3:** Overview of the components required to for the manipulator. Two servos are used and are connected in a chain with the other parts.

**Figure 2.5:** A 3D-render of the manipulator without the servo in the first joint. The servo horn is to be attached to the mount on top of the image.

A 3D-render of the manipulator is displayed in Fig. 2.5. Here, only the servo of joint 2 is seen. This is due to the fact that the servo of joint 1 is placed inside the servo cage of the quadrotor. Further descriptions of the manipulator geometry such as the Denavit-Hartenberg parameters are presented in Chapter 3.

## 2.3 Payload Pallet

As a payload for the pick-and-place tasks a simple pallet with a large loop for the manipulator's end-effector to grip was designed. The pallet is simply a cuboid-shaped, mostly hollow object in which it is possible to place objects. When using drones for transporting materiel, it is likely that objects would be placed in or on standardized containers, whether it is a standard UIC 435-2 (EUR/EPAL-pallet), a propietary carrier board in a manufacturing environment, or specialized containers for blood samples, medicine, etc.

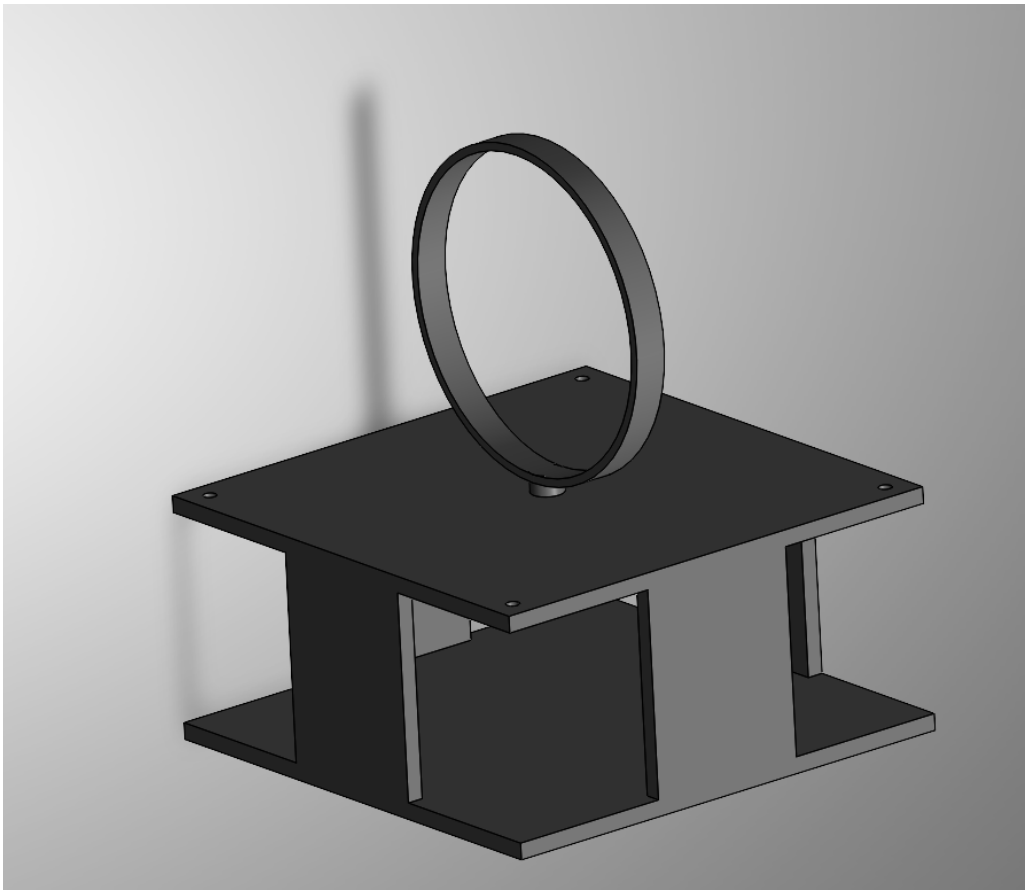


**Figure 2.6:** A 3D-render of the payload as seen with a large loop on top for the hook to catch.

When using the UAM to pick up the payload it should be able identify the pallet. One way to achieve this is with visual markers such as ArUco markers or AprilTags that are visible to the onboard camera. The ArUco marker (see Fig. 2.7) is chosen for this work due to its good pose estimation accuracy, detection rate, and comparatively low computational cost for detecting single markers [15].

**Figure 2.7:** ArUco marker visual representation. Imagine this will be glued onto the payload shown in Figure 2.6. It can be placed either onto the sides of the payload, or on the top part next to the loop.

# 3 | Modelling

In this chapter, a mathematical model of the system in question will be formulated. The first section will describe the quadrotor kinematic model alone, followed by a section describing the manipulator model. The third section is dedicated to combining these two previous models into a coupled model and deriving the dynamics of the resulting system. The final dynamic modelling is based on Lagrangian mechanics. The kinematic and dynamic modelling largely follows the approach outlined in [16]. The fourth section contains a Newton-Euler approach to modelling the UAM which was initially attempted without success. The last section will describe models for sensors, which will be used to calibrate the sensors and for developing measurement models for state estimation.

## 3.1 Quadrotor Model

Throughout this section the model of a quadrotor is developed. The quadrotor consists of four motors mounted on arms extending from a central body, as described in the previous chapter. The modelling of the quadrotor will begin with attaching coordinate frames to the aircraft and describing the chosen parametrization of the frame rotations before moving on to the individual actuator models. Finally these will be used to construct the kinematic model of the quadrotor.

### 3.1.1 Coordinate Frames

To describe the quadrotor's pose and motion two different frames of reference are used, namely the inertial frame and the body frame. The inertial frame is the frame in which Newton's laws are valid and is considered fixed. The body frame on the other hand is attached to a point on the aircraft and moves with the quadrotor. As the inertial frame a local fixed frame with origin at ground level is chosen and is denoted $\{\mathcal{W}\}$. The body frame, denoted $\{B_0\}$ with axes $\{x_B, y_B, z_B\}$, is attached as seen in Fig. 3.1 with its origin located in the center of mass (CoM) of the quadrotor. Furthermore, a link frame denoted $\{L_0\}$ is attached such that it coincides with the body frame. The base link frame of the quadrotor $\{L_0\}$ and the body frame $\{B_0\}$ are the same frame in this work, but for the sake of generality, are treated as if they were separate. The point of attaching both a link and a body frame becomes more clear, once the manipulator model also comes into play. This is delayed until section 3.2.

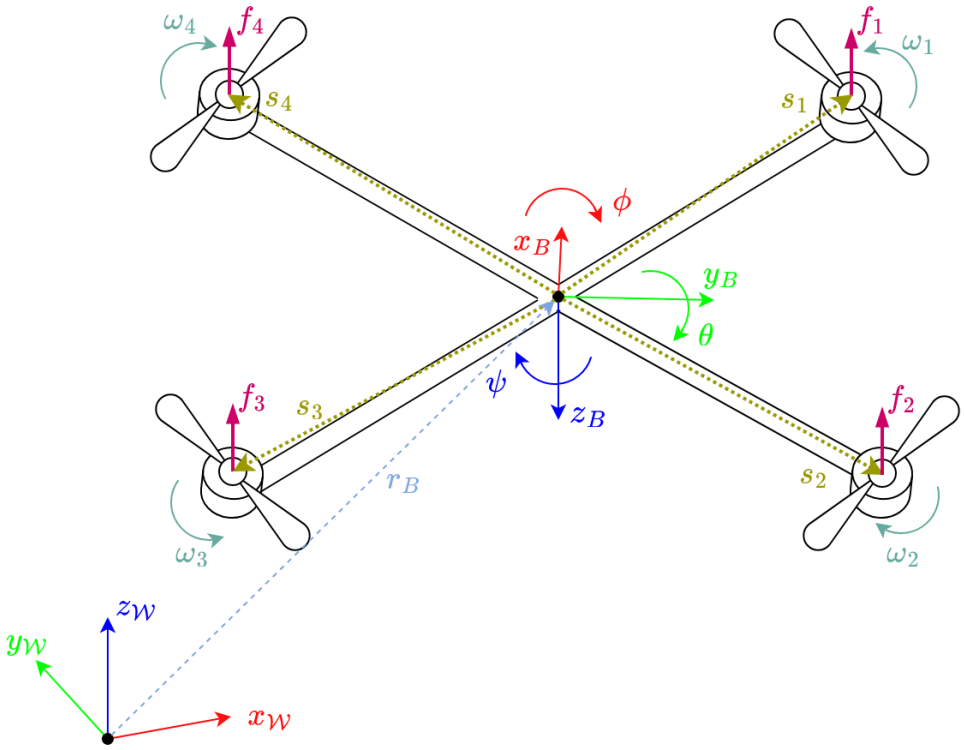



**Figure 3.1:** Attachment of frames, with direction of rotation, actuator torques and forces, and relevant position vectors. Here, the body axis $x_B$ should be imagined as pointing away from the reader due to the perspective

An alternative to this attachment is to let the body $x_B$- and $y_B$-axes point along the rotor arms rather than between them. Since it is more efficient to produce pure roll or pitch movements using two rotors rather than one, this attachment is advantageous, as the alternative attachment would decouple the two rotors along the axis of the assigned rotor arm from the rotational dynamics around that very same axis. It can be argued that an automatic flight controller can easily handle this i.e. by a simple frame transformation, but attaching the frame like this, improves model clarity at no significant cost.

### 3.1.2 Attitude Parametrization

Rotations between the inertial frame and the body frame can be parametrized in multiple different ways. The rotation matrix (or attitude matrix) $\mathbf{R} \in SO(3)$ is a typical representation defined as:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \tag{3.1}$$

and the rotation from the body frame to the world frame is denoted ${}^{\mathcal{W}}\mathbf{R}_B$. Rotation matrices are orthonormal, meaning they exhibit the following properties:

$$\mathbf{R}^{-1} = \mathbf{R}^T \quad \Rightarrow \quad \mathbf{R}\mathbf{R}^T = \mathbf{I}_3, \tag{3.2a}$$

$$\|r_i\|_2 = 1 \quad \forall\, i = 1, 2, 3, \tag{3.2b}$$

where $r_i$ denotes column $i$ in $\mathbf{R}$. This makes the rotation matrix a 9-parameter description of the attitude with a total of 6 constraints. However, other parametrizations with fewer parameters exist. One such parametrization are the Euler angles (or Tait-Bryan angles). Here, an angle for the rotation around each axis is given. The order of rotation can be chosen arbitrarily. A typical rotation order is the order $X$-$Y$-$Z$ or alternatively $Z$-$Y$-$X$. Let $(\phi,\ \theta,\ \psi)$ denote the angles for rotations around the $x$-, $y$-, and $z$-axis (or *roll, pitch,* and *yaw* (RPY) angles respectively), then, the equivalent rotation matrices for each rotation is given by:

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{bmatrix}, \tag{3.3a}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \tag{3.3b}$$

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.3c}$$

which when chained as extrinsic rotations with the $X$-$Y$-$Z$ order, combines as

$$\mathbf{R}_{xyz} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{3.4a}$$

$$= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{3.4b}$$

This provides the equivalent rotation matrix. The RPY or Euler-angle parametrization is intuitive and has the lowest amount of parameters necessary to fully parametrize $SO(3)$, the group of rotations in Euclidean 3D-space $\mathbb{R}^3$. The major downside to this parametrization however, is that it is not free of singularities. That is to say, there is a point at which the parametrization breaks down and loses a degree of freedom. This occurs when the second angle in the rotation order is exactly 90 degrees, as this causes the third rotation, to rotate in the same plane as the first rotation did. This problem is known as gimbal lock. The loss of a degree of freedom in the parametrization can cause serious issues for a controller, potentially causing a UAV to crash. Although, it is not expected to be a typical occurence for a UAM to be tilted at a 90° angle, it may occur under extreme circumstances, and could make it significantly more difficult to correct the aircraft's attitude.

One way to get around this issue is to use a different parametrization, such as the *attitude quaternion*. The attitude quaternion is a unit quaternion, i.e. a quaternion

with unit norm. Quaternions are a group of hypercomplex numbers with 1 real and 3 complex dimensions defined as:

$$q = q_0 + q_1 i + q_2 j + q_3 k, \tag{3.5a}$$
$$i^2 = j^2 = k^2 = -1, \tag{3.5b}$$
$$ij = -ji = k, \tag{3.5c}$$
$$jk = -kj = i, \tag{3.5d}$$
$$ki = -ik = j. \tag{3.5e}$$

The intuition for the quaternion parametrization is related to the axis-angle representation and the geometric interpretation of complex numbers, as it is a way to encode the same information. The axis-angle representation represents a rotation in 3D-space as a vector $\hat{\mathbf{u}}$ and a corresponding angle of rotation $\theta$ about that vector. Rotations in the $xy$-plane (around the $z$-axis) can be described by a complex number by applying Euler's formula:

$$e^{i\theta} = \cos\theta + i\sin\theta. \tag{3.6}$$

To extend this complex representation to three dimensions, an extra complex component for both the $x$-, and $y$-axes are added. Euler's formula can then be extended to the quaternions as:

$$q = e^{\frac{\theta}{2}(\hat{u}_x i + \hat{u}_y j + \hat{u}_z k)} = \cos\left(\frac{\theta}{2}\right) + \sin\left(\frac{\theta}{2}\right)(\hat{u}_x i + \hat{u}_y j + \hat{u}_z k). \tag{3.7}$$

The factor of $\frac{1}{2}$ ensures that the unit length constraint of the attitude quaternion is satisfied.
It is common to use a vector notation which allows for the use of linear algebra techniques such as matrix-vector products for quaternion operations:

$$\mathbf{q} = \begin{bmatrix} q_0 \\ \mathbf{q}_{1:3} \end{bmatrix}, \quad \mathbf{q}_{1:3} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}. \tag{3.8}$$

With this notation, the real part of the quaternion $q_0$ is typically referred to as the *scalar part* and $\mathbf{q}_{1:3}$ is referred to as the *vector part*. Rotations are applied by quaternion multiplication. Two competing quaternion multiplication operators exist [17], the original being introduced by Hamilton ($\odot$) and the alternative by Shuster ($\otimes$). Given two quaternions $\mathbf{q}$ and $\mathbf{p}$ the operators are defined as:

$$\mathbf{p} \odot \mathbf{q} = \begin{bmatrix} p_0 q_0 - \mathbf{p}_{1:3}^T \mathbf{q}_{1:3} \\ q_0 \mathbf{p}_{1:3} + p_0 \mathbf{q}_{1:3} + \mathbf{p}_{1:3} \times \mathbf{q}_{1:3} \end{bmatrix}, \tag{3.9a}$$
$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_0 q_0 - \mathbf{p}_{1:3}^T \mathbf{q}_{1:3} \\ q_0 \mathbf{p}_{1:3} + p_0 \mathbf{q}_{1:3} - \mathbf{p}_{1:3} \times \mathbf{q}_{1:3} \end{bmatrix}, \tag{3.9b}$$

where the operator $\times$ should be interpreted as the vector cross product. The only difference between $\odot$ and $\otimes$ is a sign flip at the cross product and it follows that:

$$\mathbf{p} \odot \mathbf{q} = \mathbf{q} \otimes \mathbf{p}. \tag{3.10}$$

Sequences of rotations are therefore performed by multiplying each subsequent quaternion from the right (Hamilton's quaternion product) or the left (Shuster's quaternion product). Throughout this work, Hamilton's quaternion convention will be applied. To rotate a vector $\mathbf{v} \in \mathbb{R}^3$ by a quaternion $\mathbf{q}$, the quaternion and its conjugate $\mathbf{q}^*$ (its inverse $\mathbf{q}^{-1}$ for non-unit quaternions) is multiplied on the opposite sides of $\mathbf{v}$:

$$\begin{bmatrix} 0 \\ \mathbf{v}' \end{bmatrix} = \mathbf{q} \odot \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \odot \mathbf{q}^\star = \begin{bmatrix} 0 \\ \mathbf{R}(\mathbf{q})\mathbf{v} \end{bmatrix}, \tag{3.11}$$

and $\mathbf{R}(\mathbf{q})$ is the equivalent rotation matrix given by:

$$\mathbf{R}(\mathbf{q}) = \left(q_0^2 - \|\mathbf{q}_{1:3}\|^2\right)\mathbf{I}_3 - 2q_0\left[\mathbf{q}_{1:3}\right]_\times + 2\mathbf{q}_{1:3}\mathbf{q}_{1:3}^T, \tag{3.12}$$

where the skew-symmetric cross product matrix is defined as:

$$\left[\mathbf{x}\right]_\times \triangleq \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}, \quad \mathbf{x} \in \mathbb{R}^3. \tag{3.13}$$

It can be shown that the vector rotations above are equivalent to applying Rodrigues' rotation formula, which is more computationally efficient, and is given by:

$$\mathbf{v}' = \mathbf{v} + 2\mathbf{q}_{1:3} \times \left(\mathbf{q}_{1:3} \times \mathbf{v} + q_0\mathbf{v}\right). \tag{3.14}$$

Lastly, let $\vec{\omega}$ be the angular velocities in the body frame $\{B\}$, then the kinematics of the quaternion representation are given by:

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{q} \odot \vec{\omega} = \frac{1}{2}\begin{bmatrix} -\mathbf{q}_{1:3}^T \\ q_0\mathbf{I}_3 + \left[\mathbf{q}_{1:3}\right]_\times \end{bmatrix}\vec{\omega} \triangleq \frac{1}{2}\Xi(\mathbf{q})\vec{\omega}. \tag{3.15}$$

At a later stage, these kinematics will need to be used in their discrete-time form which is given by:

$$\mathbf{q}_{k+1} = \Omega(\vec{\omega}_k)\mathbf{q}_k, \tag{3.16a}$$

$$\Omega(\vec{\omega}_k) = \begin{bmatrix} \cos\left(\frac{1}{2}\|\vec{\omega}_k\|\Delta t\right) & -\boldsymbol{\psi}_k^T \\ \boldsymbol{\psi}_k & \cos\left(\frac{1}{2}\|\vec{\omega}_k\|\Delta t\right) I_3 - \left[\boldsymbol{\psi}_k\right]_\times \end{bmatrix}, \tag{3.16b}$$

$$\psi_k = \frac{\sin\left(\frac{1}{2}\|\vec{\omega}_k\|\Delta t\right)}{\|\vec{\omega}_k\|}\vec{\omega}_k, \tag{3.16c}$$

where $\Delta t$ is the time step.
Regarding the notation $\mathbf{q} \odot \vec{\omega}$ in Eq. 3.15: Although $\vec{\omega}$ is a vector in $\mathbb{R}^3$, and not a quaternion or vector in $\mathbb{R}^4$ this is a common way to denote a quaternion product where the scalar part of the "$\vec{\omega}$ quaternion" is zero. That is, the three component vector is treated as a quaternion with its scalar part equal to zero (also known as a *pure quaternion*). [18]

### 3.1.3 Quadrotor Kinematics

The kinematics of the quadrotor are developed in the inertial frame to have a common frame of reference once the quadrotor and manipulator models are combined

and the dynamics are to be derived. Due to the amount of frames, there are many transformations to keep track of. The notation for the homogeneous transformation mapping from a frame $\{B\}$ to another frame $\{A\}$ is:

$$ {}^{A}\mathbf{T}_{B} = \begin{bmatrix} {}^{A}\mathbf{R}_{B} & {}^{A}\mathbf{r}_{B}^{A} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.17} $$

where ${}^{A}\mathbf{R}_{B} \in \mathbb{R}^{3\times3}$ is the rotation matrix from $\{B\}$ to $\{A\}$, and ${}^{A}\mathbf{r}_{B}^{A}$ is the vector between the frames. Here, the prefix ${}^{A}(\cdot)$ denotes the frame from which the vector is observed, and the postscripts $(\cdot)_{B}^{A}$ denote that the vector extends from the origin of $\{A\}$ to the origin of $\{B\}$.
The pose of the quadrotor is described by its orientation $\mathbf{q}$ and the position of the link frame $\{L_0\}$ with respect to the inertial frame $\{\mathcal{W}\}$, denoted ${}^{\mathcal{W}}\mathbf{r}_{L_0}^{\mathcal{W}}$. The homogeneous transformation from the quadrotor to the inertial frame is given by:

$$ {}^{\mathcal{W}}\mathbf{T}_{L_0} = \begin{bmatrix} {}^{\mathcal{W}}\mathbf{R}_{L_0} & {}^{\mathcal{W}}\mathbf{r}_{L_0}^{\mathcal{W}} \\ \mathbf{0} & 1 \end{bmatrix}, \tag{3.18} $$

where ${}^{\mathcal{W}}\mathbf{R}_{L_0} = \mathbf{R}(\mathbf{q})$ is the rotation matrix associated with the attitude quaternion $\mathbf{q}$. The velocity of the quadrotor frame in the inertial frame ${}^{\mathcal{W}}\dot{\mathbf{r}}_{L_0}^{\mathcal{W}}$ is the time derivative of the position vector. The quaternion kinematics are related to the angular velocity relative to the inertial frame as seen from the inertial frame by:

$$ \begin{bmatrix} 0 \\ {}^{\mathcal{W}}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} \end{bmatrix} = 2 \begin{bmatrix} q_0 & -\mathbf{q}_{1:3}^{T} \\ \mathbf{q}_{1:3} & q_0\mathbf{I}_3 - [\mathbf{q}_{1:3}]_{\times} \end{bmatrix}^{T} \dot{\mathbf{q}} \triangleq 2\,[\mathbf{q}]_{\otimes}^{T}\,\dot{\mathbf{q}}, \tag{3.19} $$

or if only considering the non-zero part:

$$ {}^{\mathcal{W}}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} = 2\begin{bmatrix} -\mathbf{q}_{1:3} & q_0\mathbf{I}_3 + [\mathbf{q}_{1:3}]_{\times} \end{bmatrix}\dot{\mathbf{q}} \triangleq 2\,\Psi(\mathbf{q})^{T}\dot{\mathbf{q}}. \tag{3.20} $$

The angular velocity in the inertial frame can be obtained from the local frame angular velocity by:

$$ {}^{\mathcal{W}}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} = {}^{\mathcal{W}}\mathbf{R}_{L_0}\,{}^{L_0}\boldsymbol{\omega}_{L_0}^{\mathcal{W}}. \tag{3.21} $$

Lastly, relating the angular velocity relative to the inertial frame as seen from the local frame to $\dot{\mathbf{q}}$ is described by Eq. 3.15, but can be rearranged as:

$$ {}^{L_0}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} = 2\,\Xi(\mathbf{q})^{T}\dot{\mathbf{q}}. \tag{3.22} $$

### 3.1.4 Actuator Thrust & Torque

The quadrotor is equipped with actuators in the form of rotors. Here, the relation between the rotor inputs and the resulting forces and torques is described. In Fig. 3.1 the motor rotation and thrust forces are denoted as $(\omega_i,\, f_i)\ \forall\ i \in \{1,2,3,4\}$ and with the vectors $s_i\ \forall\ i \in \{1,2,3,4\}$ denoting the position of the $i$th motor in the body frame $\{B_0\}$. The body frame is attached at the CoM around which the actuators are not symmetrically positioned, necessitating the use of the individual position vectors. All motors and propellers are of the same brand and model and are

therefore assumed to be identical. The thrust force $f_i$ generated by a rotor spinning with angular velocity $\omega_i$ is given by:

$$f_i = k_t \omega_i^2, \tag{3.23}$$

where $k_t$ is the lift constant. The thrust is directed along the rotor's axis of rotation, aligning with the body frame $z_B$-axis. However, the input is not rotor speed but a PWM-signal $u_i \in [0, 100]$. The relationship between the PWM-signal and the rotor speed is modelled as approximately linear (since the voltage across the motor is directly proportional to the rotation speed) leading to:

$$\omega_i \approx k_r u_i \quad \Rightarrow \quad f_i \approx k_t (k_r u_i)^2. \tag{3.24}$$

The vector $\mathbf{f}_i$ is the thrust force vector expressed in the body frame which can be written as:

$$\mathbf{f}_i = \begin{bmatrix} 0 \\ 0 \\ f_i \end{bmatrix}. \tag{3.25}$$

The drag on the propeller, along with the rotor inertia results in a torque being applied around the rotor axis:

$$\tau_i = k_\tau \omega_i^2 + J_r \dot{\omega}_i, \tag{3.26}$$

where $k_\tau$ is the drag constant and $J_r$ is the moment of inertia of the rotor. The contribution of the term $J_r \dot{\omega}_i$ is typically small enough to be disregarded leaving the approximation:

$$\tau_i \approx k_\tau (k_r u_i)^2. \tag{3.27}$$

Furthermore, the thrust force at each rotor creates a torque in the body frame. When combined with $\tau_i$ the total torque from each rotor is given by:

$$\boldsymbol{\tau}_i = \mathbf{s}_i \times \mathbf{f}_i + \begin{bmatrix} 0 \\ 0 \\ \tau_i \end{bmatrix}, \tag{3.28}$$

where $\boldsymbol{\tau}_i \in \mathbb{R}^3$ is the torque exerted on the quadrotor by rotor $i$, $\mathbf{s}_i$ is the position vector of rotor $i$. The total actuation forces expressed in the body frame are therefore given by:

$$\mathbf{f}_t = \sum_{i=1}^{4} \mathbf{f}_i, \tag{3.29}$$

$$\boldsymbol{\tau}_t = \sum_{i=1}^{4} \boldsymbol{\tau}_i. \tag{3.30}$$

## 3.2 Manipulator Model

Here, the kinematics of the manipulator with 2 links are described. The assumption of a fixed base is made, where the fixed base will have the frame $\{L_0\}$. The first step is to attach the frames at various places on the body. These frames determine the forward kinematic equations of the manipulator.

### 3.2.1 Frames & Denavit-Hartenberg Parameters

In this section joint frames, link frames, and body frames will be attached at various places on the body of the manipulator. The Denavit-Hartenberg (DH) parameters (as described in [19]) will be used for attaching the link frames $\{L_i\}$ for all links $i = 1, \ldots, N_L$. The joint frames $\{\mathcal{J}_j\}$ are simply attached such that they are located in the center of joints $j = 1, \ldots, N_J$, and have the same orientation as the frame of link $j$ when the joint is in its zero-position. Similarly, the body frames $\{B_i\}$, $i = 1, \ldots, N_B$ are attached in the center of mass of each link, with the same orientation as the respective link frames. The Denavit-Hartenberg parameters are given in Table 3.1. An explanation of the parameters is given below:

- $\alpha_i$ is the angle from $z_{L_i}$ to $z_{L_{i+1}}$ measured about $x_{L_i}$,
- $a_i$ is the distance from $z_{L_i}$ to $z_{L_{i+1}}$ measured along $x_{L_i}$,
- $d_i$ is the distance from $x_{L_{i-1}}$ to $x_{L_i}$ measured along $z_{L_i}$,
- $\theta_i$ is the angle from $x_{L_{i-1}}$ to $x_{L_i}$ measured about $z_{L_i}$.

Given that in the DH-parameter convention the $z$-axis of a link coincides with the corresponding joint axis, $\theta_i$ is the joint angle for revolute joints. The procedure for attaching the frames is outlined in [19]. The attached link frames are displayed in Fig. 3.2. The positions of the CoM of each link is determined from the CAD model. The transformation between $\{B_i\}$ and $\{L_i\}$ is fixed and does not vary. The joint frames do not move as their joints rotate, so the transformation between $\{\mathcal{J}_i\}$ and $\{L_{i-1}\}$ is also fixed and the joint positions can likewise be determined in the CAD model of the manipulator.

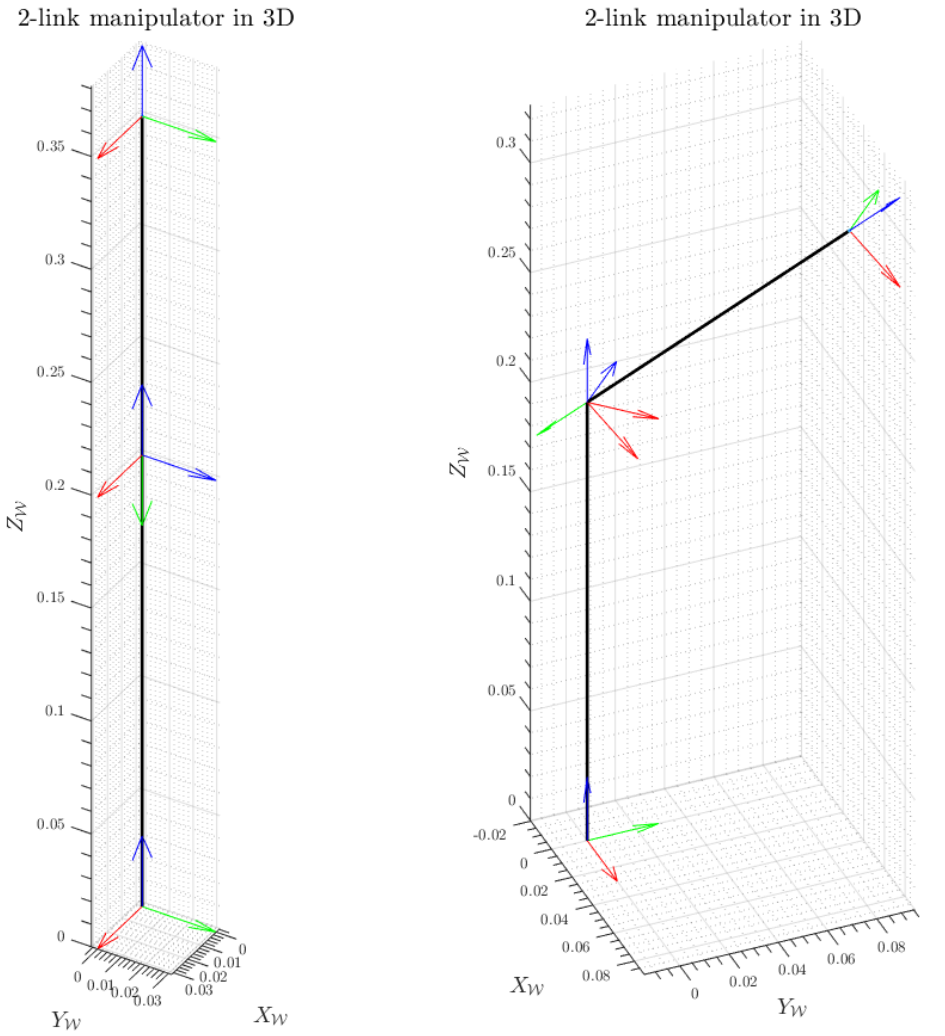
| $i$ | $\alpha_{i-1}$ | $a_{i-1}$ | $d_i$ | $\theta_i$ |
|---|---|---|---|---|
| 1 | 0 | 0 | $l_0 + l_1$ | $\theta_1$ |
| 2 | $-\frac{\pi}{2}$ | 0 | 0 | $\theta_2$ |
| 3 | $\frac{\pi}{2}$ | 0 | $l_2$ | 0 |

**Table 3.1:** Table of DH-parameters. Although, the manipulator only has 2 links, a third is included here, denoting the end-effector frame. Furthermore, the quantities $l_0$, $l_1$, and $l_2$ denote the distances from the base (link 0) to joint 1, from joint 1 to link 1, and from link 2 to the end effector respectively.

### 3.2.2 Forward Kinematics

The forward kinematics of a manipulator with link frames attached according the DH-parameter convention can be described as a chain of homogeneous transformations from each link to the previous link [19]. The individual transformations are given by:

$$ {}^{L_{i-1}}\mathbf{T}_{L_i} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -d_i\sin(\alpha_{i-1}) \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & d_i\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.31} $$




**(a)** The manipulator in its zero position.

**(b)** The manipulator where both joint angles are 45°.

**Figure 3.2:** Visualization of attached link frames for the robot in two different configurations. At the base of the manipulator (bottom of the plot), the frame $\{L_0\}$ is seen. The two frames $\{L_1\}$ and $\{L_2\}$ have the same origin but with different orientations. At the end of the second link, the end-effector frame is located.

Then the transformation from any link $i$ to the base link is given by:

$$^{L_0}\mathbf{T}_{L_i} = \prod_{j=1}^{i} {}^{L_{j-1}}\mathbf{T}_{L_j}, \tag{3.32}$$

where the individual transformation matrices are *right-multiplied*. Furthermore, using CAD data the homogeneous transformations $^{L_{i-1}}\mathbf{T}_{\mathcal{J}_i}$, $^{\mathcal{J}_i}\mathbf{T}_{L_i}$ and $^{L_i}\mathbf{T}_{B_i}$ can be found. With these transformations, it is possible to find the pose of any frame in the system relative to any other by a series of matrix multiplications of the transformation matrices.

### 3.2.3 Velocities and Jacobians

Now the velocities of each body (both angular and translational) and their respective Jacobians are described. The Jacobians will become useful when deriving the dynamics of the system [16]. The angular velocity of a body is the same everywhere on the body, meaning that the joints rotate at the same angular velocity as the entire body. Since only the joints are rotating, the angular velocity of a body relative to the base link is given by the sum of the angular velocity of its corresponding joint and all the angular velocities of bodies/joints further up the kinematic chain (i.e. the bodies closer to the base):

$$^{L_0}\boldsymbol{\omega}_{B_i}^{L_0} = \sum_{j=0}^{i} {}^{L_0}Z_{\mathcal{J}_j}\dot{\theta}_j, \tag{3.33}$$

where $^{L_0}Z_{\mathcal{J}_j}$ is the $z$-axis (and therefore the joint axis) of joint $\mathcal{J}_j$ as seen from the base frame, and $\dot{\theta}_j$ is the joint velocity of joint $\mathcal{J}_j$. The joint axis $^{L_0}Z_{\mathcal{J}_j}$ can be easily determined from the rotational part of the transformation $^{L_0}\mathbf{T}_{\mathcal{J}_j}$:

$$^{L_0}Z_{\mathcal{J}_j} = {}^{L_0}\mathbf{R}_{\mathcal{J}_j} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}. \tag{3.34}$$

The rotational Jacobians for the bodies are given by:

$$^{L_0}\mathbf{J}_{\omega,i} = \begin{bmatrix} \dfrac{\partial\, {}^{L_0}\boldsymbol{\omega}_{B_i}^{L_0}}{\partial\dot{\theta}_1} & \cdots & \dfrac{\partial\, {}^{L_0}\boldsymbol{\omega}_{B_i}^{L_0}}{\partial\dot{\theta}_{N_J}} \end{bmatrix}, \tag{3.35}$$

which is easily derived due to the linearity of Eq. 3.33 in the joint velocities $\dot{\boldsymbol{\theta}} = \begin{bmatrix}\dot{\theta}_1, \ldots, \dot{\theta}_{N_J}\end{bmatrix}^T$. This also means that $^{L_0}\boldsymbol{\omega}_{B_i}^{L_0} = {}^{L_0}\mathbf{J}_{\omega,i}\dot{\boldsymbol{\theta}}$.
For the translational velocities of the fixed base manipulator, the velocity of the CoM (and by extension the body frame) relative to the base depends only on the angular velocities of the bodies. The translational velocity of a body $\{B_i\}$ is caused by the rotation of all joints $\{\mathcal{J}_j\}$ further up the kinematic chain (meaning all $j \leq i$)

as seen from the base link is given by:

$$
\begin{aligned}
{}^{L_0}\dot{\mathbf{r}}_{B_i}^{\mathcal{J}_j} &= {}^{L_0}Z_{\mathcal{J}_j}\dot{\theta}_j \times \left({}^{L_0}\mathbf{r}_{\mathcal{J}_j}^{L_0} - {}^{L_0}\mathbf{r}_{B_i}^{L_0}\right) && \text{(3.36a)}\\
&= {}^{L_0}Z_{\mathcal{J}_j}\dot{\theta}_j \times {}^{L_0}\mathbf{r}_{B_i}^{\mathcal{J}_j} && \text{(3.36b)}\\
&= \left({}^{L_0}Z_{\mathcal{J}_j} \times {}^{L_0}\mathbf{r}_{B_i}^{\mathcal{J}_j}\right)\dot{\theta}_j. && \text{(3.36c)}
\end{aligned}
$$

Getting the velocity of each body $\{B_i\}$ relative to the base link is now a matter of summing all the velocity components from all joints contributing to the velocity of each body:

$$
{}^{L_0}\dot{\mathbf{r}}_{B_i}^{L_0} = \sum_{j=1}^{i} {}^{L_0}\dot{\mathbf{r}}_{B_i}^{\mathcal{J}_j}. \tag{3.37}
$$

The Jacobian for the translational velocities of each body is then given by:

$$
{}^{L_0}\mathbf{J}_{t,i} = \begin{bmatrix} \dfrac{\partial\, {}^{L_0}\dot{\mathbf{r}}_{B_i}^{L_0}}{\partial\dot{\theta}_1} & \cdots & \dfrac{\partial\, {}^{L_0}\dot{\mathbf{r}}_{B_i}^{L_0}}{\partial\dot{\theta}_{N_J}} \end{bmatrix}, \tag{3.38}
$$

which again due to the linearity of Eq. 3.36c is easy to determine and therefore ${}^{L_0}\dot{\mathbf{r}}_{B_i}^{L_0} = {}^{L_0}\mathbf{J}_{t,i}\dot{\boldsymbol{\theta}}$.

### 3.2.4 Servo Model

As a last step in the modelling of the manipulator, the servos are described. The Dynamixel servos used in the manipulator have built-in controllers for several different purposes including a current controller. Under the assumption of perfect controller tracking the proportional relation between torque and current of a DC-motor can be applied directly. The motor torque $\tau_m$ is given by [19]:

$$
\tau_m = k_m i_a, \tag{3.39}
$$

where $i_a$ is the armature current and $k_m$ is the torque constant.

## 3.3 Aerial Manipulator Model

In order to derive the dynamic model of the UAM in the Euler-Lagrange framework, first the kinematics of the manipulator and the quadrotor must be combined. This involves treating the quadrotor as a 6 DoF joint, attached to the base link of the manipulator.

### 3.3.1 Kinematics

The kinematics of the UAM are made by introducing an extra transformation ${}^{\mathcal{W}}\mathbf{T}_{L_0}$ between the inertial frame $\{\mathcal{W}\}$ and the quadrotor/base link $\{L_0\}$. The transformation is given by the quadrotor kinematics of Eq. 3.18. Due to this transformation,

the pose and velocities of each manipulator link must also be changed. The position of each CoM in the inertial frame is given by:

$$ {}^{\mathcal{W}}\mathbf{r}_{B_i}^{\mathcal{W}} = {}^{\mathcal{W}}\mathbf{r}_{L_0}^{\mathcal{W}} + {}^{\mathcal{W}}\mathbf{r}_{B_i}^{L_0}, \tag{3.40} $$

where the position of each body relative to the base link as seen in the inertial frame ${}^{\mathcal{W}}\mathbf{r}_{B_i}^{L_0}$ is found by the following rotation of ${}^{L_0}\mathbf{r}_{B_i}^{L_0}$:

$$ {}^{\mathcal{W}}\mathbf{r}_{B_i}^{L_0} = \mathbf{q} \odot \begin{bmatrix} 0 & {}^{L_0}\mathbf{r}_{B_i}^{L_0 T} \end{bmatrix}^T \odot \mathbf{q}^* = {}^{\mathcal{W}}\mathbf{R}_{L_0}\, {}^{L_0}\mathbf{r}_{B_i}^{L_0}. \tag{3.41} $$

The translational velocity of the bodies can be found by taking the time derivative of Eq. 3.40. The full derivation can be found in [16] and only the final result is stated here:

$$ {}^{\mathcal{W}}\dot{\mathbf{r}}_{B_i}^{\mathcal{W}} = {}^{\mathcal{W}}\mathbf{r}_{L_0}^{\mathcal{W}} + {}^{\mathcal{W}}\mathbf{R}_{L_0}\, {}^{L_0}\mathbf{J}_{t,i}\dot{\boldsymbol{\theta}} - 2\,{}^{\mathcal{W}}\mathbf{R}_{L_0} \left[ {}^{L_0}\mathbf{r}_{B_i}^{L_0} \right]_\times \Xi(\mathbf{q})^T \dot{\mathbf{q}}. \tag{3.42} $$

The angular velocity of each body in the world frame is the sum of the angular velocity of the base link (Eq. 3.19) and the angular velocity of the body relative to the base link. Since this is parametrized by the quaternion, the result is a pure quaternion with its vector part being the angular velocity:

$$ \begin{bmatrix} 0 \\ {}^{\mathcal{W}}\boldsymbol{\omega}_{B_i}^{\mathcal{W}} \end{bmatrix} = \begin{bmatrix} 0 \\ {}^{\mathcal{W}}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} \end{bmatrix} + \mathbf{q} \odot \begin{bmatrix} 0 \\ {}^{L_0}\boldsymbol{\omega}_{B_i}^{L_0} \end{bmatrix} \odot \mathbf{q}^* \tag{3.43} $$

$$ = 2\,[\mathbf{q}]_\otimes^T \dot{\mathbf{q}} + \begin{bmatrix} 0 \\ {}^{\mathcal{W}}\mathbf{R}_{L_0}\, {}^{L_0}\mathbf{J}_{\omega,i} \end{bmatrix} \dot{\boldsymbol{\theta}}. \tag{3.44} $$

### 3.3.2 System Jacobians & Lagrangian Dynamics

Using the kinematic equations the equations of motion can be derived with Lagrangian mechanics [16]. First, the state $\mathbf{x}$ and the generalized coordinates $\boldsymbol{\xi}$ are written:

$$ \mathbf{x} = \begin{bmatrix} \boldsymbol{\xi}^T & \dot{\boldsymbol{\xi}}^T \end{bmatrix}^T, \tag{3.45} $$

$$ \boldsymbol{\xi} = \begin{bmatrix} \mathbf{r}^T & \mathbf{q}^T & \boldsymbol{\theta}^T \end{bmatrix}^T \in \mathbb{R}^{7+N_j}. \tag{3.46} $$

The generalized coordinates must be chosen such that they are a complete and mutually independent representation of the system. The choice of generalized coordinates are a complete representation of the pose of the system, but the quaternion coordinates are not independent of each other due to the unit norm requirement. This can be alleviated by adding a constraint which produces a constraint force. The constraint forces will be discussed in more detail in section 3.3.3. The velocities of each body in the UAM parametrized by the generalized coordinates can be written as:

$$ \mathbf{v} = \begin{bmatrix} {}^{\mathcal{W}}\dot{\mathbf{r}}_{B_0}^{\mathcal{W}\,T},\ \dots\ ,{}^{\mathcal{W}}\dot{\mathbf{r}}_{B_{N-1}}^{\mathcal{W}}{}^{T},\ \begin{bmatrix} 0 & {}^{\mathcal{W}}\boldsymbol{\omega}_{B_0}^{\mathcal{W}\,T} \end{bmatrix}^T,\ \dots\ ,\ \begin{bmatrix} 0 & {}^{\mathcal{W}}\boldsymbol{\omega}_{B_{N-1}}^{\mathcal{W}}{}^{T} \end{bmatrix}^T \end{bmatrix}^T, \tag{3.47} $$

which implies that there exists a Jacobian ${}^{\mathcal{W}}\mathbf{J}$ of the system with respect to the generalized coordinates such that $\mathbf{v} = {}^{\mathcal{W}}\mathbf{J}\dot{\boldsymbol{\xi}}$. This full system Jacobian is given by:

$$ {}^{\mathcal{W}}\mathbf{J} = \begin{bmatrix} {}^{\mathcal{W}}\mathbf{J}_{t,0} \\ \vdots \\ {}^{\mathcal{W}}\mathbf{J}_{t,N_J} \\ {}^{\mathcal{W}}\mathbf{J}_{\omega,0} \\ \vdots \\ {}^{\mathcal{W}}\mathbf{J}_{\omega,N_J} \end{bmatrix} \in \mathbb{R}^{7N_B \times (7+N_J)}, \tag{3.48} $$

where the translational and rotational parts are found as:

$$ {}^{\mathcal{W}}\mathbf{J}_{t,i} = \begin{bmatrix} I_3 & | & -2{}^{\mathcal{W}}\mathbf{R}_{L_0} \left[ {}^{L_0}\mathbf{r}_{B_i}^{L_0} \right]_{\times} \Xi(\mathbf{q})^T & | & {}^{\mathcal{W}}\mathbf{R}_{L_0} \, {}^{L_0}\mathbf{J}_{t,i} \end{bmatrix} \in \mathbb{R}^{3 \times (7+N_J)}, \tag{3.49} $$

$$ {}^{\mathcal{W}}\mathbf{J}_{\omega,i} = \begin{bmatrix} 0_{4\times 3} & | & 2\left[\mathbf{q}\right]_{\otimes}^T & | & \begin{bmatrix} 0 \\ {}^{\mathcal{W}}\mathbf{R}_{L_0} \, {}^{L_0}\mathbf{J}_{\omega,i} \end{bmatrix} \end{bmatrix} \in \mathbb{R}^{4 \times (7+N_J)}. \tag{3.50} $$

To derive the equations of motion, one approach is to determine the kinetic and potential energies $E_{\text{kin}}$ and $E_{\text{pot}}$, and defining the Lagrangian [16], [20]:

$$ \mathcal{L} = E_{\text{kin}} - E_{\text{pot}}. \tag{3.51} $$

The energies can be found with:

$$ E_{\text{kin}} = \frac{1}{2} \dot{\boldsymbol{\xi}}^T \, {}^{\mathcal{W}}\mathbf{J}^T \mathbf{M}_L \, {}^{\mathcal{W}}\mathbf{J} \dot{\boldsymbol{\xi}}, \tag{3.52} $$

$$ E_{\text{pot}}\left(\boldsymbol{\xi}\right) = \sum_{i=0}^{N_{B-1}} m_i \mathbf{g}_0 \, {}^{\mathcal{W}}\mathbf{r}_{B_i}^{\mathcal{W}}, \tag{3.53} $$

where the matrix $\mathbf{M}_L$ is the matrix with masses and inertias of the links on the diagonal:

$$ \mathbf{M}_L = \text{blockdiag}\left( \mathbf{I}_3 m_0, \ \ldots \ , \ \mathbf{I}_3 m_{N_B-1}, \ {}^{\mathcal{W}}\Theta_0, \ldots, \ {}^{\mathcal{W}}\Theta_{N_B-1} \right), \tag{3.54} $$

where $\Theta$ is the matrix:

$$ {}^{\mathcal{W}}\Theta_i = \begin{bmatrix} v & \mathbf{0} \\ \mathbf{0} & {}^{\mathcal{W}}\Phi_i \end{bmatrix} \in \mathbb{R}^{4\times 4}, \tag{3.55} $$

which has an arbitrary positive scalar $v$ and ${}^{\mathcal{W}}\Phi_i$, the actual inertia tensor of body $i$ expressed in the world frame, on its diagonal. The local body frame inertia tensor is related to the inertial frame inertia tensor by [20]:

$$ {}^{\mathcal{W}}\Phi_i = {}^{\mathcal{W}}\mathbf{R}_{B_i} \, {}^{B_i}\Phi_i \, {}^{\mathcal{W}}\mathbf{R}_{B_i}^T. \tag{3.56} $$

With the energies determined, the equations of motion are then given by:

$$ \frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{\xi}}} - \frac{\partial \mathcal{L}}{\partial \boldsymbol{\xi}} = \mathbf{f}_{\boldsymbol{\xi}} + \mathbf{f}_c, \tag{3.57} $$

where $\mathbf{f}_{\boldsymbol{\xi}}$ are the generalized external forces acting on the generalized coordinates, and $\mathbf{f}_c$ are constraint forces, that is they are virtual forces that keep any constraints in check (the quaternion unit norm constraint). This can be written in the standard form:

$$\mathbf{M}(\boldsymbol{\xi})\,\ddot{\boldsymbol{\xi}} + \mathbf{C}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}})\dot{\boldsymbol{\xi}} + \mathbf{g}(\boldsymbol{\xi}) = \mathbf{f}_{\boldsymbol{\xi}} + \mathbf{f}_c, \tag{3.58}$$

where $\mathbf{M}(\boldsymbol{\xi}) \in \mathbb{R}^{N_x \times N_x}$ is the system mass matrix, $\mathbf{C}(\boldsymbol{\xi}, \dot{\boldsymbol{\xi}}) \in \mathbb{R}^{N_x \times N_x}$ is the system's Coriolis matrix, and $\mathbf{g}(\boldsymbol{\xi})$ is the gravitational term. Obtaining the matrices $\mathbf{M}$ and $\mathbf{C}$ and the vector $\mathbf{g}$ from the expression resulting from evaluating Eq. 3.57 can be unwieldy, especially for large expressions. The closed form expression for the mass matrix $\mathbf{M}$ can be obtained using the system Jacobian:

$$\mathbf{M} = {}^{\mathcal{W}}\mathbf{J}^T \mathbf{M}_L \, {}^{\mathcal{W}}\mathbf{J}. \tag{3.59}$$

Using the closed form expression of the mass matrix, the Coriolis matrix can be determined using the Christoffel symbols (see [20]). Each entry in the matrix is given by:

$$\mathbf{C}_{ij} = \frac{1}{2} \sum_{k=1}^{N_x} \left( \frac{\partial \mathbf{M}_{ij}}{\partial \boldsymbol{\xi}_k} + \frac{\partial \mathbf{M}_{ik}}{\partial \boldsymbol{\xi}_j} + \frac{\partial \mathbf{M}_{jk}}{\partial \boldsymbol{\xi}_i} \right) \dot{\boldsymbol{\xi}}_k. \tag{3.60}$$

The gravitational terms are comparatively simple to obtain because the potential energy is typically a far simpler expression than the kinetic energy.

$$\mathbf{g} = \nabla E_{\text{pot}} = \frac{\partial E_{\text{pot}}}{\partial \boldsymbol{\xi}} \tag{3.61}$$

The expressions can grow rapidly in size for each additional link added to the manipulator. For the UAM in question the resulting mass matrix and Coriolis matrix each contain over 24000 and 249000 operations respectively. For this reason the closed form expressions are not stated here.

In order to obtain the equations of motion, it is necessary to understand how to map the actuation forces to the generalized forces. These are not the same, since the actuation forces are the body frame thrust, the body frame torques and the joint torques, but the generalized coordinates are the inertial frame positions, the quaternion, and the joint positions. The joint torques act directly on their corresponding generalized coordinates so their mapping is simply the identity. The quadrotor body frame forces act on the body frame coordinates, which map to the inertial frame through the rotation matrix ${}^{\mathcal{W}}\mathbf{R}_{L_0}$. Mapping the torques to quaternion forces is not quite as straightforward. One approach to derive the map is to apply the principle of virtual work as in [21], where the author shows that the mapping is given as:

$$\mathbf{f}_{\mathbf{q}} = 2\,\Xi(\mathbf{q})\tau_t, \tag{3.62}$$

where $\mathbf{f}_{\mathbf{q}}$ is the generalized quaternion force. The force map $\mathbf{M}_f$ is then defined as:

$$\mathbf{M}_F = \text{blockdiag}\left({}^{\mathcal{W}}\mathbf{R}_{L_0},\ 2\,\Xi(\mathbf{q}),\ \mathbf{I}_{N_J}\right), \tag{3.63}$$

which leads to the relation:

$$\mathbf{f}_{\boldsymbol{\xi}} = \mathbf{M}_F \, {}^{L_0}\mathbf{f}_B, \tag{3.64}$$

where $\mathbf{f}_{\boldsymbol{\xi}}$ is the generalized force vector, and ${}^{L_0}\mathbf{f}_B$ is the vector of local body forces containing the total thrust force $\mathbf{f}_t$, the quadrotor torques $\boldsymbol{\tau}_t$, and the joint torques $\boldsymbol{\tau}_{\mathcal{J}}$:

$$ {}^{L_0}\mathbf{f}_B = \begin{bmatrix} \mathbf{f}_t^T & \boldsymbol{\tau}_t^T & \boldsymbol{\tau}_{\mathcal{J}}^T \end{bmatrix}^T . \tag{3.65} $$

This leaves only the constraints and the corresponding constraint forces to fully determine the equations of motion.

### 3.3.3 Constraint Forces

The quaternion unit norm constraint is a *holonomic* constraint, meaning it is a constraint imposed on the position variables (here the generalized coordinates) and can be expressed in the form:

$$ \boldsymbol{\phi}(\boldsymbol{\xi}) = \mathbf{0}. \tag{3.66} $$

This is the case for the quaternion norm constraint:

$$ \|\mathbf{q}\| = 1 \quad \Leftrightarrow \quad \|\mathbf{q}\|^2 = 1 \quad \Leftrightarrow \quad \|\mathbf{q}\|^2 - 1 = 0, \tag{3.67} $$

which can also be written equivalently as:

$$ \boldsymbol{\phi}(\boldsymbol{\xi}) = \mathbf{q}^T\mathbf{q} - 1. \tag{3.68} $$

This form is preferred as it leads to a particularly simple Jacobian $\mathbf{J}_\phi$. The purpose of the Jacobian, is to write the constraint in its equivalent *Pfaffian* form, which is given by:

$$ \mathbf{J}_\phi(\boldsymbol{\xi})\,\dot{\boldsymbol{\xi}} = \mathbf{0}. \tag{3.69} $$

Then the constraint force can be written with the Lagrange multipliers, as it is common in the framework of Lagrangian mechanics [20], such that the equations of motion take the form:

$$ \mathbf{M}\ddot{\boldsymbol{\xi}} + \mathbf{C}\dot{\boldsymbol{\xi}} + \mathbf{g} - \lambda\mathbf{J}_\phi = \mathbf{f}_{\boldsymbol{\xi}}, \tag{3.70a} $$

$$ \mathbf{J}_\phi\dot{\boldsymbol{\xi}} = \mathbf{0}, \tag{3.70b} $$

where $\lambda$ is the vector of Lagrange multipliers. The goal is to obtain the forward dynamics of the UAM, which in the unconstrained case ($\mathbf{f}_c = \mathbf{0}$) can be achieved by solving Eq. 3.58 for $\ddot{\boldsymbol{\xi}}$:

$$ \ddot{\boldsymbol{\xi}} = \mathbf{M}^{-1}\left(\mathbf{f}_{\boldsymbol{\xi}} - \mathbf{C}\dot{\boldsymbol{\xi}} - \mathbf{g}\right). \tag{3.71} $$

In the constrained case the system of equations in Eq. 3.70 must be solved. Since only the effect of the constraint force is of interest and the constrant force itself is not important, a choice is made opt not to compute it explicitly. Instead the Lagrange multipliers can be eliminated by a procedure explained in [20]. The idea is to determine a projection matrix which projects the generalized forces onto the components that do work on the system. The projection matrix will have rank $N_x - k$, where $N_x$ is the number of generalized coordinates and $k$ is the number

of constraints. The procedure is begun by taking the time derivative of Eq. 3.70b which yields:

$$\mathbf{J}_\phi(\boldsymbol{\xi})\,\ddot{\boldsymbol{\xi}} + \dot{\mathbf{J}}_\phi(\boldsymbol{\xi})\,\dot{\boldsymbol{\xi}} = \mathbf{0}. \tag{3.72}$$

Solving Eq. 3.70a for $\ddot{\boldsymbol{\xi}}$ and substituting into Eq. 3.72 gives:

$$\dot{\mathbf{J}}_\phi\dot{\boldsymbol{\xi}} + \mathbf{J}_\phi\mathbf{M}^{-1}\left(\mathbf{f}_{\boldsymbol{\xi}} + \mathbf{J}_\phi^T\lambda - \mathbf{C}\dot{\boldsymbol{\xi}} - \mathbf{g}\right) = \mathbf{0}. \tag{3.73}$$

This expression can be solved for $\lambda$:

$$\boldsymbol{\lambda} = \left(\mathbf{J}_\phi\mathbf{M}^{-1}\mathbf{J}_\phi^T\right)^{-1}\left(-\dot{\mathbf{J}}_\phi\dot{\boldsymbol{\xi}} + \mathbf{J}_\phi\mathbf{M}^{-1}\left(\mathbf{C}\dot{\boldsymbol{\xi}} + \mathbf{g} - \mathbf{f}_{\boldsymbol{\xi}}\right)\right), \tag{3.74}$$

and from Eq. 3.72 the relation $\mathbf{J}_\phi(\boldsymbol{\xi})\,\ddot{\boldsymbol{\xi}} = -\dot{\mathbf{J}}_\phi(\boldsymbol{\xi})\,\dot{\boldsymbol{\xi}}$ can be exploited to yield (after a bit of work):

$$\left(\mathbf{I} - \mathbf{J}_\phi^T\left(\mathbf{J}_\phi\mathbf{M}^{-1}\mathbf{J}_\phi^T\right)^{-1}\mathbf{J}_\phi\mathbf{M}^{-1}\right)\left(\mathbf{M}\ddot{\boldsymbol{\xi}} + \mathbf{C}\dot{\boldsymbol{\xi}} + \mathbf{g} - \mathbf{f}_{\boldsymbol{\xi}}\right) = \mathbf{0}. \tag{3.75}$$

With the following definition:

$$\mathbf{P}_u \triangleq \mathbf{I} - \mathbf{J}_\phi^T\left(\mathbf{J}_\phi\mathbf{M}^{-1}\mathbf{J}_\phi^T\right)^{-1}\mathbf{J}_\phi\mathbf{M}^{-1}, \tag{3.76}$$

Eq. 3.75 can be written in the form:

$$\mathbf{P}_u\left(\mathbf{M}\ddot{\boldsymbol{\xi}} + \mathbf{C}\dot{\boldsymbol{\xi}} + \mathbf{g}\right) = \mathbf{P}_u\mathbf{f}_{\boldsymbol{\xi}}. \tag{3.77}$$

Here, $\mathbf{P}_u$ is a matrix that projects the generalized forces onto the components that do work on the system. Eq. 3.77 corresponds to the constrained inverse dynamics of the aerial manipulator. Making the following definition:

$$\mathbf{P} \triangleq \mathbf{M}^{-1}\mathbf{P}_u\mathbf{M} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{J}_\phi^T\left(\mathbf{J}_\phi\mathbf{M}^{-1}\mathbf{J}_\phi^T\right)^{-1}\mathbf{J}_\phi, \tag{3.78}$$

the projection matrix $\mathbf{P}$ is the projection from motions in generalized coordinates to motions that satisfies the constraint of Eq. 3.68. This gives the constrained forward dynamics of the aerial manipulator:

$$\mathbf{P}\ddot{\boldsymbol{\xi}} = \mathbf{P}\mathbf{M}^{-1}\left(\mathbf{f}_{\boldsymbol{\xi}} - \mathbf{C}\dot{\boldsymbol{\xi}} - \mathbf{g}\right). \tag{3.79}$$

These are the final equations of motion, and the forward dynamics can be used for simulating the system. Some simulations of the system using this model can be found in chapter 6.

## 3.4 Newton-Euler Aerial Manipulator Dynamics

Initially, an approach to modelling the UAM based on the Newton-Euler equations of motion was attempted. This approach is described in this section. The procedure is to model the dynamics of the two subsystems (quadrotor and manipulator) separately, before combining them. The Newton-Euler approach can result in more compact expressions, making it potentially less expensive in a computational sense [19] than the Euler-Lagrange formulation. This was the rationale for attempting it.

### 3.4.1 Quadrotor Dynamics

Here, the dynamics of the quadrotor are shortly described. The rotational dynamics are in general given by Euler's equation of rotational motion:

$$ {}^{L_0}\mathbf{\Phi}_0 \, {}^{L_0}\dot{\boldsymbol{\omega}}_{L_0}^{\mathcal{W}} = -{}^{L_0}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} \times {}^{L_0} \, \mathbf{\Phi}_0 \, {}^{L_0}\boldsymbol{\omega}_{L_0}^{\mathcal{W}} + \boldsymbol{\tau}_t \tag{3.80} $$

where ${}^{L_0}\mathbf{\Phi}_0$ is the local body frame inertia tensor. The translational dynamics in the body frame are given by Newton's equation of motion:

$$ m_0 \, {}^{L_0}\ddot{\mathbf{r}}_{L_0}^{\mathcal{W}} = {}^{\mathcal{W}}\mathbf{R}_{L_0}^T \begin{bmatrix} 0 \\ 0 \\ m_0 g \end{bmatrix} + \mathbf{f}_t - {}^{L_0} \, \boldsymbol{\omega}_{L_0}^{\mathcal{W}} \times {}^{L_0}\dot{\mathbf{r}}_{L_0}^{\mathcal{W}}, \tag{3.81} $$

where, on the right hand side of the equation, the first term accounts for the effect of gravitation (with gravitational acceleration $g$), the second term accounts for the effect of the rotor thrust which is modelled to align with the body frame $z_B$-axis. The third term accounts for the centrifugal force. When expressed in the inertial frame the centrifugal force is nullified, so the acceleration becomes:

$$ m_0 \, {}^{\mathcal{W}}\ddot{\mathbf{r}}_{L_0}^{\mathcal{W}} = \begin{bmatrix} 0 \\ 0 \\ m_0 g \end{bmatrix} + {}^{\mathcal{W}}\mathbf{R}_{L_0}\mathbf{f}_t \tag{3.82} $$

### 3.4.2 Manipulator Dynamics

The inverse dynamics of a manipulator in the Newton-Euler framework can be determined using the *Recursive Newton-Euler algorithm* (RNE), which will only be described briefly here. This algorithm starts by propagating accelerations from the base forward through the kinematic chain until the last link. This way the accelerations and velocities of each link in the chain is determined. Then, starting from the end-effector, the link forces and torques are propagated back through the kinematic chain. This way the joint torques can be determined. The algorithm is summarized in a general form in Eqs. 3.83-3.84 as it is written by [19].

#### Recursive Newton-Euler Algorithm

Forward iterations: $i \; : \; 0 \rightarrow (N_L - 1)$

$$ {}^{i+1}\boldsymbol{\omega}_{i+1} = {}^{i+1}\mathbf{R}_i \, {}^{i}\boldsymbol{\omega}_i + \dot{\theta}_{i+1} \, {}^{i+1}Z_{i+1}, \tag{3.83a} $$

$$ {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} = {}^{i+1}\mathbf{R}_i \, {}^{i}\dot{\boldsymbol{\omega}}_i + {}^{i+1}\mathbf{R}_i \, {}^{i}\boldsymbol{\omega}_i \times \dot{\theta}_{i+1} \, {}^{i+1}Z_{i+1} + \ddot{\theta}_{i+1} \, {}^{i+1}Z_{i+1}, \tag{3.83b} $$

$$ {}^{i+1}\dot{\boldsymbol{v}}_{i+1} = {}^{i+1}\mathbf{R}_i \left( {}^{i}\dot{\boldsymbol{\omega}}_i \times {}^{i}\mathbf{r}_{i+1} + {}^{i}\boldsymbol{\omega}_i \times \left( {}^{i}\boldsymbol{\omega}_i \times {}^{i}\mathbf{r}_{i+1} \right) + {}^{i}\dot{\boldsymbol{v}}_i \right), \tag{3.83c} $$

$$ \begin{aligned} {}^{i+1}\dot{\boldsymbol{v}}_{B_{i+1}} = {} & {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} \times {}^{i+1}\mathbf{r}_{B_{i+1}} \\ & + {}^{i+1}\boldsymbol{\omega}_{i+1} \times \left( {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{i+1}\mathbf{r}_{B_{i+1}} \right) + {}^{i+1}\dot{\boldsymbol{v}}_{i+1}, \end{aligned} \tag{3.83d} $$

$$ {}^{i+1}\mathbf{F}_{i+1} = m_{i+1} \, {}^{i+1}\dot{\boldsymbol{v}}_{B_{i+1}}, \tag{3.83e} $$

$$ {}^{i+1}\mathbf{N}_{i+1} = {}^{B_{i+1}}\mathbf{\Phi}_{i+1} \, {}^{i+1}\dot{\boldsymbol{\omega}}_{i+1} + {}^{i+1}\boldsymbol{\omega}_{i+1} \times {}^{B_{i+1}}\mathbf{\Phi}_{i+1} \, {}^{i+1}\boldsymbol{\omega}_{i+1}. \tag{3.83f} $$

Backward iterations: $i \ : \ N_L \rightarrow 1$

$$ {}^{i}\mathbf{f}_i = {}^{i}\mathbf{R}_{i+1}\, {}^{i+1}\mathbf{f}_{i+1} + {}^{i}\mathbf{F}_i, \tag{3.84a} $$

$$ {}^{i}\mathbf{n}_i = {}^{i}\mathbf{N}_i + {}^{i}\mathbf{R}_{i+1}\, {}^{i+1}\mathbf{n}_{i+1} + {}^{i}\mathbf{r}_{B_i} \times {}^{i}\mathbf{F}_i + {}^{i}\mathbf{r}_{i+1} \times {}^{i}\mathbf{R}_{i+1}\, {}^{i+1}\mathbf{f}_{i+1}, \tag{3.84b} $$

$$ \tau_i = {}^{i}\mathbf{n}_i^T\, {}^{i}Z_i. \tag{3.84c} $$

Here, the link frames are only denoted by their index number $i$. Furthermore the notation $(\cdot)_{B_i}$ refers the body frame located at the center of mass of link $i$. The meaning of the quantities are as follows:

- ${}^{i}\boldsymbol{\omega}_i$ is the angular velocity of link $i$,
- ${}^{i}\dot{\boldsymbol{\omega}}_i$ is the angular acceleration of link $i$,
- $\dot{\theta}_i$ is the joint velocity of joint $i$,
- $\ddot{\theta}_i$ is the joint acceleration of joint $i$,
- ${}^{i}\dot{\boldsymbol{v}}_i$ is the acceleration of the link $i$ frame,
- ${}^{i}\dot{\boldsymbol{v}}_{B_i}$ is the acceleration of the CoM of link $i$,
- ${}^{i}\mathbf{F}_i$ is the force acting on the CoM of link $i$,
- ${}^{i}\mathbf{N}_i$ is the torque acting on the CoM of link $i$,
- $m_i$ is the mass of link $i$,
- ${}^{B_i}\boldsymbol{\Phi}_i$ is the inertia tensor of link $i$ with respect to its CoM $\{B_i\}$,
- ${}^{i}\mathbf{f}_i$ is the force acting at joint $i$,
- ${}^{i}\mathbf{n}_i$ is the three-dimensional torque acting at joint $i$,
- $\tau_i$ is the joint torque (torque around the joint axis) at joint $i$.

Gravitation is included by setting ${}^{0}\dot{\boldsymbol{v}}_0 = -\bar{\mathbf{g}}$ where $\bar{\mathbf{g}}$ is the vector of gravitational acceleration. The remaining quantities for the base link 0, can be set to 0. The algorithm can be run both numerically and symbolically depending on whether a numerical or closed form solution is desired. If the algorithm is evaluated symbolically, the resulting expression can be written in the standard form for mechanical systems:

$$ \boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{g}(\boldsymbol{\theta}). \tag{3.85} $$

In order to determine the forward dynamics, this system can be solved for $\ddot{\boldsymbol{\theta}}$, yielding:

$$ \ddot{\boldsymbol{\theta}} = \mathbf{M}(\boldsymbol{\theta})^{-1} \left( \boldsymbol{\tau} - \mathbf{C}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} - \mathbf{g}(\boldsymbol{\theta}) \right). \tag{3.86} $$

This is equivalent to the Euler-Lagrange approach to modelling the dynamics which yielded an expression of the same form.

### 3.4.3 Aerial Manipulator Dynamics

In order to use the Newton-Euler framework for modelling the UAM the RNE algorithm can be applied. To modify the manipulator system to have the quadrotor as its floating base, the approach is to set the quantities of the base link fed into the RNE algorithm as the corresponding quantities of the quadrotor (e.g. ${}^0\dot{\boldsymbol{\omega}}_0$ in the RNE is set to the angular acceleration of Eq. 3.80)[9]. This yields a set of coupled differential equations which must be solved to find the standard form of Eq. 3.86. The resulting system is of course larger as it also contains the quadrotor torques and forces. Solving this system however, is not trivial, and no correct solution was found. A simulation of this model found in chapter 6 illustrates the encountered issue. Due to this issue, the choice was made to use the Lagrangian formulation instead.

## 3.5 Sensor Models

This section describes the models for the sensors. The models are to be used for calibration and as measurement models for the state estimators.

### 3.5.1 Inertial Measurement Unit

The first sensors are the accelerometer and gyroscope, which, when collected in the same integrated circuit, are collectively known as an inertial measurement unit (IMU). The accelerometer and gyroscope are described with fairly simplistic discrete-time models, very similar to each other.

#### Gyroscope

The gyroscope model assumes that the gyroscope readings at some time $k\Delta t$ are measurements of the true angular velocity in the local frame $\boldsymbol{\omega}_k$ corrupted by a bias $\boldsymbol{\beta}_{g,k}$ and a Gaussian white noise $\boldsymbol{\eta}_k$. The bias is modelled as a random walk stochastic process, integrating Gaussian white noise.

$$\boldsymbol{\omega}_{m,k} = \boldsymbol{\omega}_k + \boldsymbol{\beta}_{g,k} + \boldsymbol{\eta}_k, \qquad \boldsymbol{\eta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_g), \tag{3.87}$$

$$\boldsymbol{\beta}_{g,k} = \boldsymbol{\beta}_{g,k-1} + \boldsymbol{\zeta}_k, \qquad \boldsymbol{\zeta}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_g). \tag{3.88}$$

The noise variables $\eta_k$ and $\zeta_k$, as white noise processes, have zero mean with covariance matrices $\mathbf{Q}_g$ and $\mathbf{W}_g$.

#### Accelerometer

Similarly the accelerometer is modelled as a measurement of the true local frame acceleration $\mathbf{a}_k$ at time $k\Delta t$, plus gravitational acceleration in the local frame, corrupted by a bias $\boldsymbol{\beta}_{a,k}$ (also modelled by a random walk process) and white noise $\boldsymbol{\epsilon}_k$.

$$\mathbf{a}_{m,k} = \mathbf{a}_k + {}^{\mathcal{W}}\mathbf{R}_{\text{acc}}\,\bar{\mathbf{g}} + \boldsymbol{\beta}_{a,k} + \boldsymbol{\epsilon}_k, \qquad \boldsymbol{\epsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_a), \tag{3.89}$$

$$\boldsymbol{\beta}_{a,k} = \boldsymbol{\beta}_{a,k-1} + \boldsymbol{\varepsilon}_k, \qquad \boldsymbol{\varepsilon}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_a). \tag{3.90}$$

Again the Gaussian stochastic variables $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\varepsilon}_k$ are zero-mean and with covariance matrices $\mathbf{Q}_a$ and $\mathbf{W}_a$.

### 3.5.2 Encoders

The encoders in the servos of the manipulator are able to very accurately measure the position of the joints. Velocity measurements in encoder systems are usually made by simply taking the difference between the position measurement at time $k\Delta t$ and time $k-1\Delta t$. Position measurements in the encoder are largely free from noise but they do have quantization error. More importantly, backlash in the servo gears and loose or elastic mechanical joinings in the servo and manipulator can corrupt measurements. These effects are not explicitly modelled as it is simply deemed unnecessarily complicated. Instead the choice of a simplistic model is made given by:

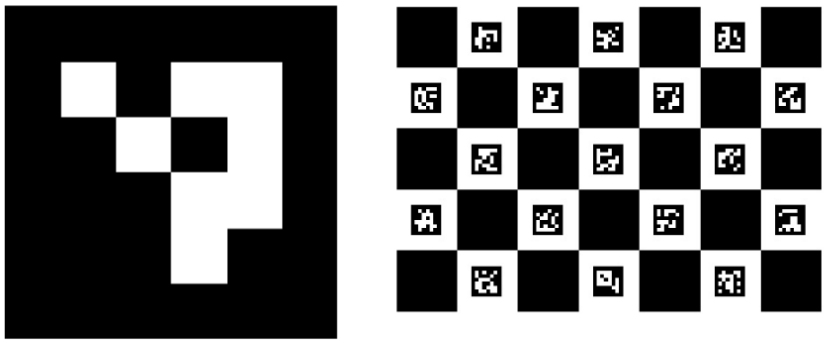
$$\theta_{m,k} = \theta_k + \upsilon_k, \qquad \upsilon_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_e), \tag{3.91}$$

where the reading $\theta_{m,k}$ is the noisy measurement of the true joint angle $\theta_k$ at time $k\Delta t$.

### 3.5.3 Camera

The camera sensor is used to detect the location of the object (to be picked by the manipulator) with respect to the camera frame. A fixed transformation from the center of the camera lens to the manipulator's gripper is then applied to the resulting coordinates.

To detect the location of an object an OpenCV library [22] that uses ChArUco and ArUco markers is used, see Fig. 3.3 for the different types of markers. The idea is to stick ArUco markers to the objects that need to be picked and detect where those markers are, and to use ChArUco markers to calibrate the camera in order to get the intrinsic camera parameters. The library then uses these parameters to relate the corners of the detected ArUco markers to the real world coordinates, which is described in more detail in section 5.1. It can do that due to the fact that the real dimensions of those markers are known.



**(a)** Standalone marker **(b)** ChArUco board

**Figure 3.3:** Types of markers: a) ArUco marker used to detect the object's location, b) ChArUco markers used to calibrate the camera

The camera is an Autofocus Camera for Raspberry Pi, and its exact model can be found in Table 2.2. The intrinsic calibration of the camera is performed at a focus distance of approx. 40 cm.

### Camera model

To understand what the camera intrinsic parameters are it is necessary to first look at the model of a pinhole camera, see Fig. 3.4.

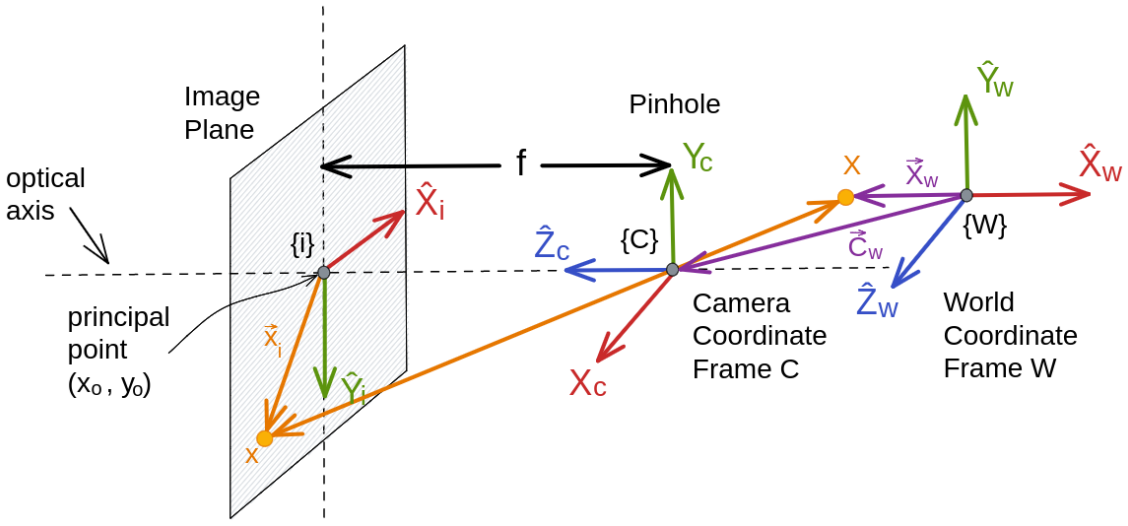



**Figure 3.4:** Pinhole camera model: f is the focal length, $\{i\}$ is the frame of the 2D image, $\{\mathcal{C}\}$ is the frame of the camera, and $\{\mathcal{W}\}$ is the world frame

The focal length is the distance between the effective central projection and the image plane. In other words it is the distance from the center of the camera lens and the point where all parallel light rays intersect. That is where the image is said to be in focus, i.e sharp. The optical axis passes through the center of the lens.

### Computing the intrinsic parameters

The OpenCV library computes a projection matrix $\mathbf{P}$, that brings the coordinates $\mathbf{X}_w$ in the world frame into the image frame coordinates $\mathbf{x}_i$:

$$\mathbf{x}_i = \mathbf{P}\mathbf{X}_w, \tag{3.92}$$

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix},$$

$$\mathbf{x}_i = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \mathbf{X}_w, \tag{3.93}$$

where $\mathbf{X}_w$ are the control points known from the ChArUco markers dimensions, and $\mathbf{x}_i$ are the control points detected from the images of the ChArUco markers, see Fig. 3.5. The size of the checker board squares' side is known, and so is the size of the smaller ArUco markers inside the white squares. The OpenCV marker detection method is using this information to calibrate the camera.

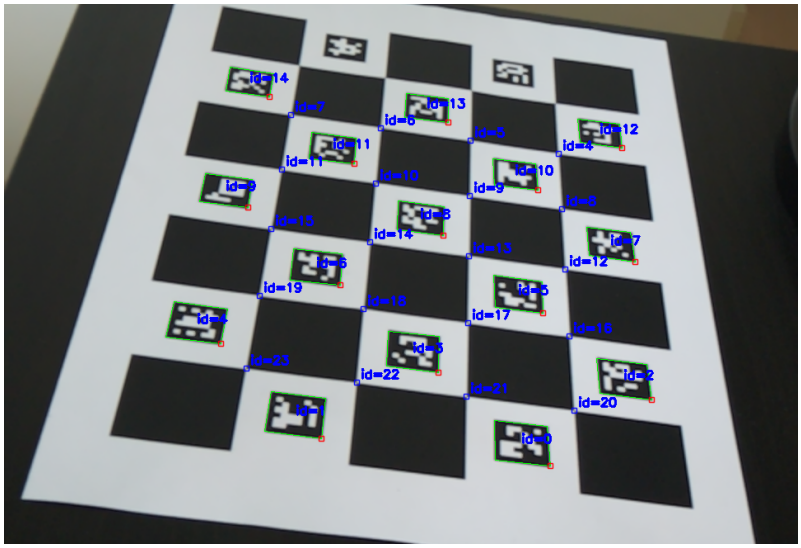



**Figure 3.5:** ChArUco control points: finding the location of the corners with respect to the image frame

Matrix $\mathbf{P}$ is then decomposed using QR factorization and other matrix computations techniques to retrieve the individual parameters:

$$\mathbf{P} = \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right], \tag{3.94}$$

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_o \\ 0 & f_y & y_o \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.95}$$

where

- $\mathbf{K}$ is called the calibration matrix,
- $f_x$ and $f_y$ are the focal lengths in pixels in the $x$ and $y$ directions,
- $x_o$ and $y_o$ are the principal point coordinates in pixels, which is located at or close to the center of the image,
- $s$ is the distortion coefficient (skew parameter), in case the image axes are not orthogonal,
- $\mathbf{R} \in \mathbb{R}^{3\times3}$ and $\mathbf{t} \in \mathbb{R}^3$ is the rotation matrix and translation vector from the world coordinate frame to the camera coordinate frame.

The focal lengths (in pixels) are computed in the following way:

$$f_x = fm_x, \tag{3.96}$$

$$f_y = fm_y, \tag{3.97}$$

where $m_x$ and $m_y$ are the pixel densities (with unit [pixels/mm]) in x and y directions and are taken from the optical sensor physical measurements, and $f$ is the focal length in millimeters.

To perform intrinsic calibration all that is needed is to have the calibration matrix $\mathbf{K}$. The rotation and translation information is only necessary when performing extrinsic calibration, which is not used in this project.

# 4 | Parameter Estimation

## 4.1 Inertia Tensors

The inertia tensor for each body in its local frame is estimated using the CAD model in SolidWorks, which has the ability to compute the inertia tensors for assembled multibody structures. This can be achieved by numerically integrating the mass density across each body. This provides the local frame inertia for each component of the assembly. To find the total inertia tensor, the local inertia tensors of the components must be moved to the same frame and rotated such that their axes align [20]. The translation is achieved by applying the *parallel axis theorem*:

$$\Phi_i' = \Phi_i + m_i \left( \mathbf{r}_i^T \mathbf{r}_i \mathbf{I}_3 - \mathbf{r}_i \mathbf{r}_i^T \right), \tag{4.1}$$

where $\Phi_i$ is the local inertia tensor of the $i$th component, $\Phi_i'$ is the translated inertia tensor, $m_i$ is the mass of the component, and $\mathbf{r}_i$ is the vector between the component body frame and the common assembly frame. To align all the inertia tensors they are rotated into the common reference frame. This can be achieved with:

$$\Phi_i'' = \mathbf{R}_i \Phi_i' \mathbf{R}_i^T, \tag{4.2}$$

as was also stated in Eq. 3.56. The total inertia tensor is then found by summation of the $N$ individual components in the specific assembly:

$$\mathbf{\Phi} = \sum_{i=1}^{N} \Phi_i'' \tag{4.3}$$

These computations are left to the CAD software. The inertia tensor and position of the CoM of the manipulator servos are both provided by the manufacturer [23].

## 4.2 Rotor Parameters

In section 3.1.4 the constants $k_r$, $k_t$, and $k_\tau$ are described. These are determined by experiment in a thrust identification stand, consisting of a two lever arms of equal length connected at a right angle. The connection point can rotate. At the end of one the rotor is attached and the other is pressing down on a scale, measuring the thrust. The rotation speed can be measured with a digital tachometer. Providing a throttle signal (PWM-signal) to the ESC makes the rotor rotate and data can be

collected. The measurements are made with a fixed supply voltage of 16V.

Thrust and angular velocity measurements were recorded along with their respective throttle inputs. A least squares fit to the actuator thrust model of Eq. 3.23 was made. The results are displayed in Fig. 4.1 (for $k_r$) and Fig. 4.2 (for $k_t$). The red curves depict the fitted models, plotted against a few selected measured data points (blue). A decrease in accuracy is noticeable at the uppermost end of the throttle range, but at the middle of the range where the system will mostly operate the fits are more accurate.

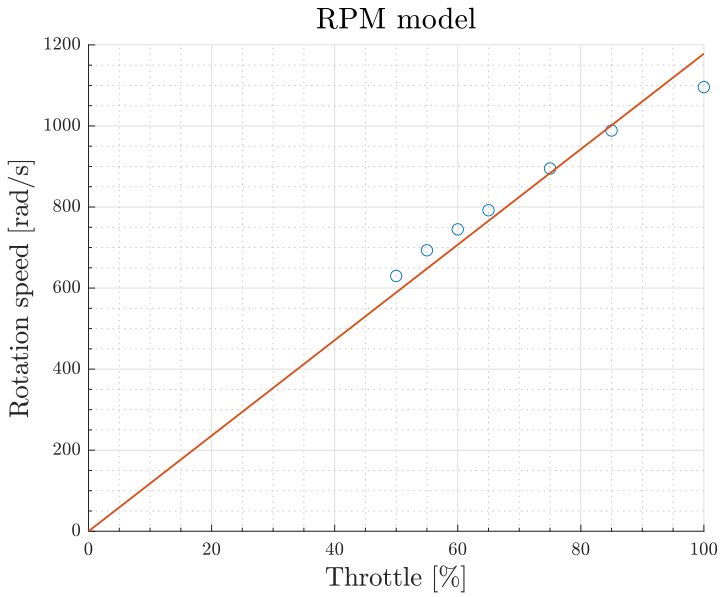



**Figure 4.1:** Illustration of the fitted linear model $\omega = k_r u$ against a few select data points.

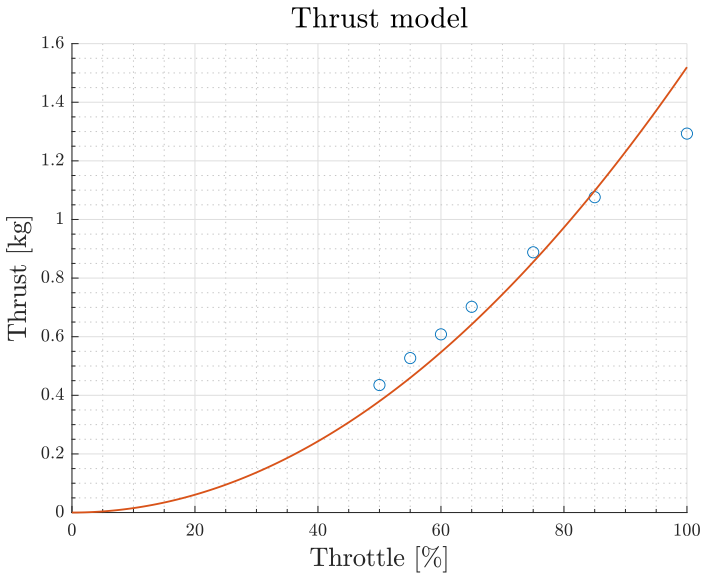



**Figure 4.2:** Illustration of the fitted quadratic model $f_t = k_t(k_r u)^2$ against a few select data points.

The fitted parameter values are found as:

$$k_r = 11.783,$$
$$k_t = 1.095 \times 10^{-6},$$

and the RMSE of the two fits and the data points are computed as:

$$\epsilon_{k_r} = 42.950,$$
$$\epsilon_{k_t} = 0.0111.$$

The manufacturer of the motor and the propeller provides torque data for the motor-propeller combination which is used to make the fit. The result is displayed in Fig. 4.3.

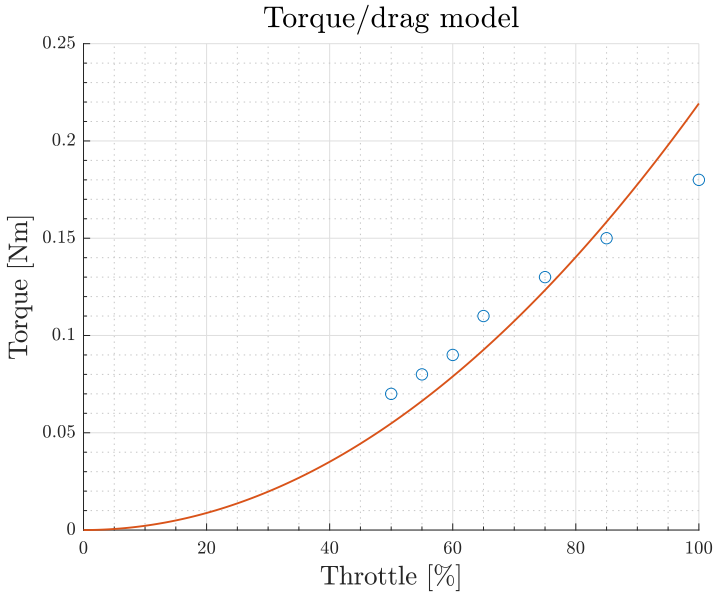



**Figure 4.3:** Illustration of the fitted quadratic model $\tau = k_\tau(k_r u)^2$ plotted against manufacturer data points.

The parameter $k_\tau$ is computed as:

$$k_\tau = 1.579 \times 10^{-7},$$

and this fit has the following RMSE:

$$\epsilon_{k_\tau} = 0.00677.$$

## 4.3 Servo Torque Constant

The servo torque constant of the servos in the manipulator links from Eq. 3.39 are estimated based on the performance curve that the manufacturer makes available [23]. The curve is sampled and a linear fit is made, which is displayed in Fig. 4.4 where the torque constant is computed as:

$$k_m = 1.402.$$

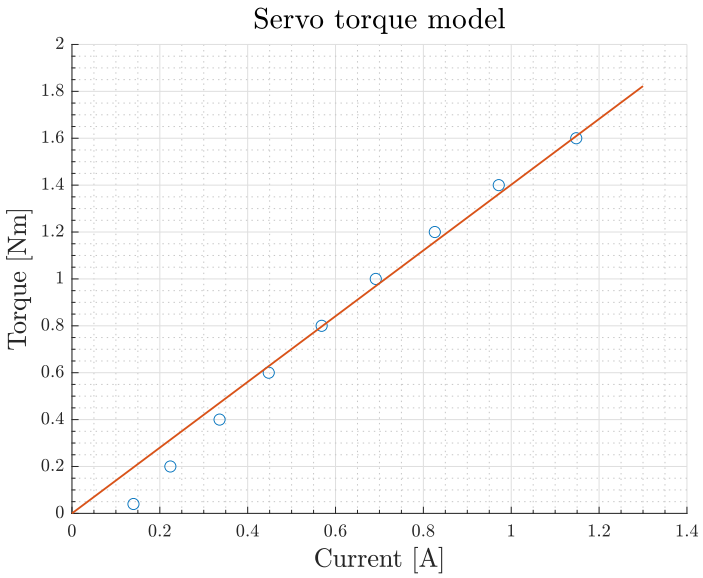



**Figure 4.4:** Illustration of the fitted linear model $\tau_m = k_m i_a$ plotted against the sampled data points from the performance graph.

While this model can be used, a fairly large error is observed at the lower end of the range caused by a deadband in the actuator. This can become a problem for accurate control. A modification of the linear model to make it affine can be made:

$$\tau_m = k_m i_a + \iota, \tag{4.4}$$

where $\iota \in \mathbb{R}$ is an offset to model the deadband. To avoid the problem of modelling negative torques with positive currents the two cases can be applied in practice:

$$\tau_m = \begin{cases} k_m i_a + \iota & \text{if } k_m i_a + \iota > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

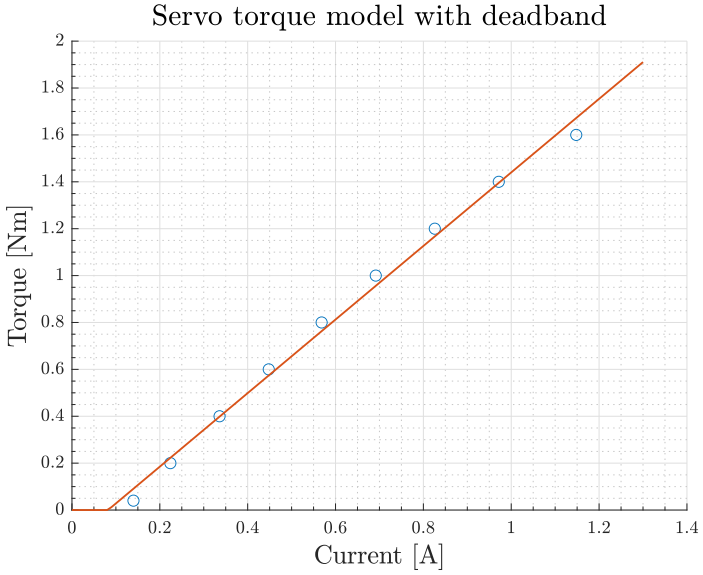



**Figure 4.5:** Illustrated of servo torque model with deadband.

The deadband model is not easy to use in many control algorithms due to its non-linear nature, so the original model of Eq. 3.39 can be used if necessary and once the desired torque is determined a mapping between the two can be applied.

## 4.4 IMU noise and bias

To estimate the noise level and more importantly the biases of the IMU, data is collected while the two sensors are at rest on a level surface. The collected data is displayed in Fig. 4.6. Here, the accelerometer data on the $z$-axis has been offset by the gravitational acceleration $g$ to make the biases more easily visible.

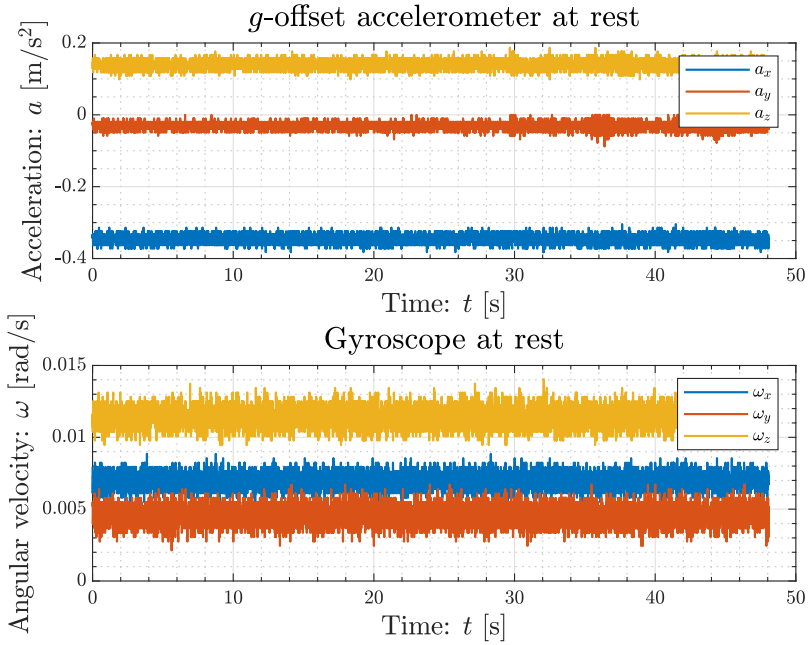



**Figure 4.6:** Accelerometer and gyroscope data for the LSM6DOX IMU at rest.

The biases can be estimated as the means of the signals in Fig. 4.6 and the variance of the noise can be estimated as the variance of the signals. This process can be repeated every time the system is powered on to provide an initial estimate for bias since it can change over time, for instance due to changes in the ambient temperature. The means can be estimated with the sample mean of the measurements $\mathbf{y}$ given by:

$$\hat{\boldsymbol{\beta}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_m \tag{4.6}$$

In Fig. 4.7 the data with the sample mean subtracted is displayed.

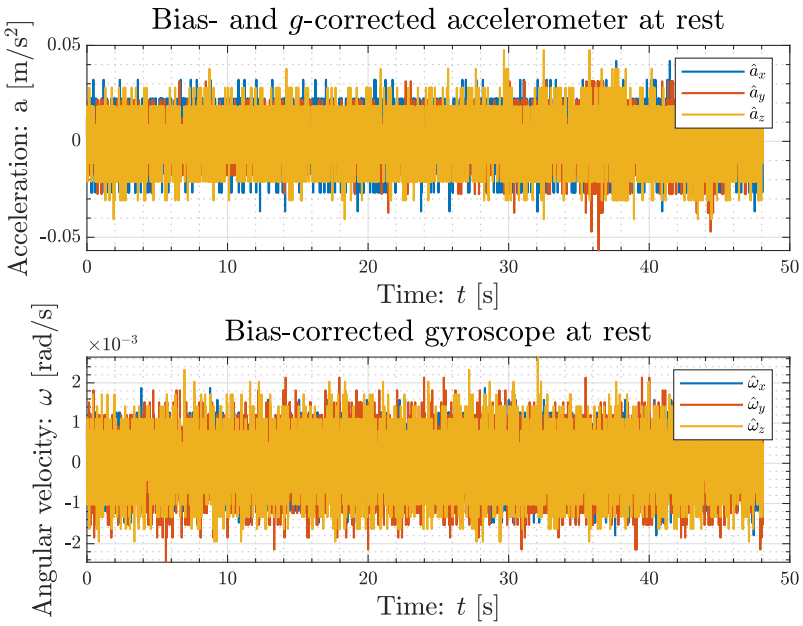



**Figure 4.7:** IMU corrected data

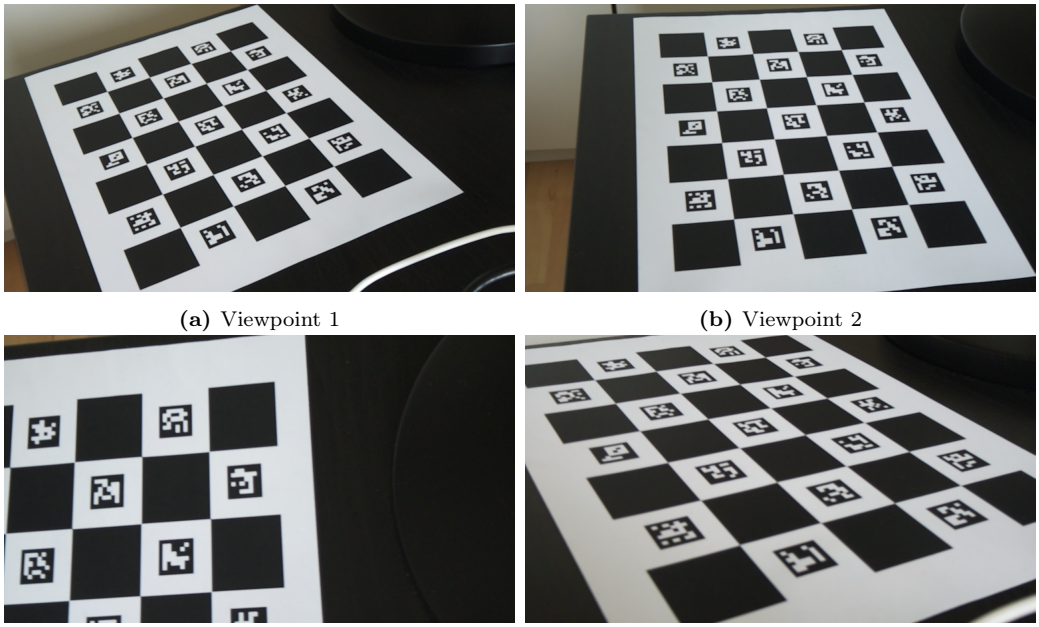
Noise variance on each channel can similarly be estimated as the sample variance:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{y} - \mu)^2 \tag{4.7}$$

where $\mu$ is the sample mean of the channel. Noise variances can also be found in the datasheet of the IMU [24]. It is assumed that the variance measurements in the datasheet are more reliable so these are used.

## 4.5 Camera Calibration

The necessary parameters to be identified are part of the camera calibration matrix $K$ described in Eq. 3.95. That is achieved by taking pictures of the ChArUco board from different viewpoints and feeding them to the OpenCV library. Occlusions are allowed, as noticed in Figs. 4.8c and 4.8d. That is due to the presence of the ArUco pattern inside the checkerboard pattern, which helps detecting which side of the pattern the identified square corners are located at.



**(a)** Viewpoint 1 **(b)** Viewpoint 2

**(c)** Viewpoint 3 **(d)** Viewpoint 4

**Figure 4.8:** ChArUco marker captured from four different viewpoints, used to calibrate the camera.

The result can be seen in Eq. 4.8, and is later used for pose estimation, as described in section 5.1.

$$\mathbf{K} = \begin{bmatrix} 1764.6 & 0 & 1182.5 \\ 0 & 1764.6 & 895.71 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

# Part II

# Estimation & Control

# 5 | State Estimation

This chapter describes the state estimator developed for the Aerial Manipulator. State estimation is necessary to create a full state feedback controller. First the measurements obtained from the onboard sensors are described, followed by a description of the state estimator itself.

## 5.1 Camera Measurements

The result of this measurement is the estimation of the ArUco marker's pose with respect to the camera frame. To achieve that, the marker(s) are first detected in 2D image space, and then the pose is estimated using the intrinsic camera parameters.

**Marker detection**

The known geometry properties of the ArUco markers are saved in a dictionary. The only saved property is the marker size in bits. Provided with this information the OpenCV *detectMarkers()* function can detect the position of the four corners in the correct order, such that they correspond to the world location of the dictionary saved corners. Saving the corner coordinates in the right order is important for achieving the right transformations necessary for detecting the ArUco marker's pose. See Fig. 5.1 for a visual representation of a control point correspondence (the purple vectors).

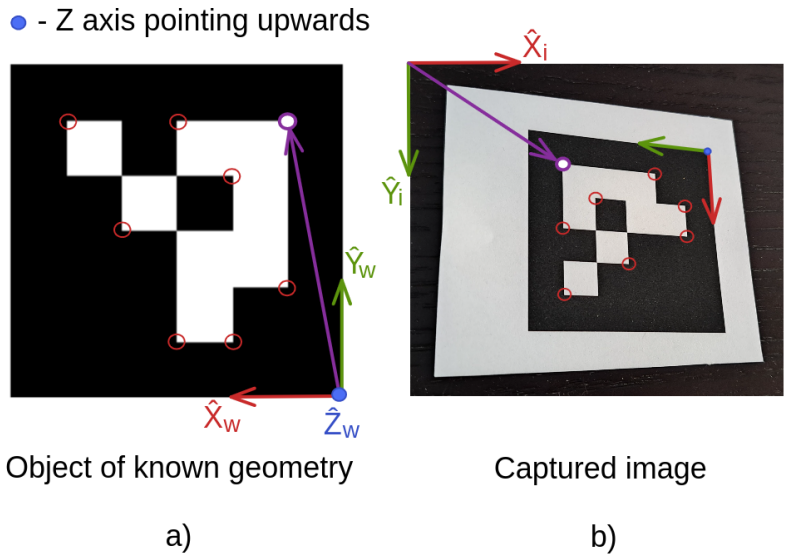



**Figure 5.1:** ArUco control points: a) finding the location of the corners in the chosen world frame b) finding the location of the corners with respect to the image frame

The OpenCV marker detection algorithm is divided in two main steps. The first one involves image segmentation that leads to contours. The contours that are not convex or do not resemble a square shape are discarded. The second step includes verification whether the detected marker is a qualified marker. This is where the inner white pattern of the marker is used. The white and black pixels are counted and compared to the information saved into the dictionary. If a match is found, a marker has been detected.

### Pose computation

To estimate the pose of the ArUco markers, the rotation and translation that minimizes the projection error from 3D to 2D correspondences is solved. The pose of a marker is placed in the center of the marker and calculated from its corner coordinates, see Fig. 5.2.

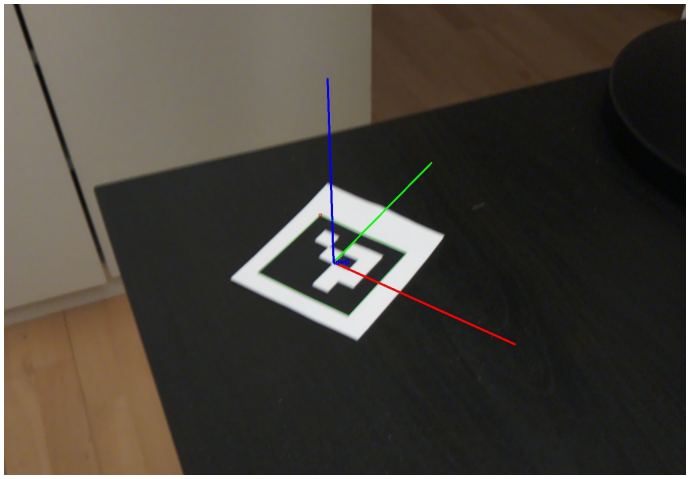


**Figure 5.2:** ArUco pose estimation, i.e. the marker's coordinate system

The necessary transformations are presented in Eqs. 5.1 through 5.3. To be more specific, Eqs. 5.1 and 5.2 describe the method of projecting 3D world coordinates onto the image plane.

$$\mathbf{x}_i = \mathbf{K}\left[\mathbf{I}|\mathbf{0}\right](\mathbf{R}\mathbf{X}_w - \mathbf{R}\mathbf{C}_w), \tag{5.1}$$

where $\mathbf{C}_w$ is the coordinates of the camera center in world frame. The rest of the parameters are explained in section 3.5.3. The detailed version of Eq. 5.1 is presented in 5.2:

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s & x_o \\ 0 & f_y & y_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{5.2}$$

The rotation matrices and translation vectors allow 3D world points to be expressed in 3D camera points:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{5.3}$$

However, the OpenCV library does not produce the rotation in the form of a rotation matrix, but in the form of a rotation vector. The rotation vector is a minimal representation of the axis-angle representation previously mentioned in section 3.1.2. It is related to the axis-angle representation by:

$$\mathbf{a}_e = \theta\hat{\mathbf{u}} \;\Rightarrow\; \|\mathbf{a}_e\| = \theta. \tag{5.4}$$

It is therefore easily mapped to a quaternion by:

$$\mathbf{q}(\mathbf{a}_e) = \begin{bmatrix} \cos\left(\frac{1}{2}\|\mathbf{a}_e\|\right) \\ \frac{\sin\left(\frac{1}{2}\|\mathbf{a}_e\|\right)}{\|\mathbf{a}_e\|}\mathbf{a}_e \end{bmatrix}. \tag{5.5}$$

Lastly, the camera frame does not coincide with the body frame of the quadrotor. There exists a fixed transformation between the two frames which is best determined using the CAD model. This transformation ${}^{B}\mathbf{T}_{\mathcal{C}}$ is applied to obtain the measurements as seen from the body frame $\{B\}$.

## 5.2 State Estimator

In this section a state estimator is described. Due to the dynamical model of the UAM being as complex as it is, evaluating the model in real-time at a rate sufficient for stabilizing a quadrotor becomes very computationally expensive. For this reason the dynamical model will not be used for the state estimator. Instead, a vision-aided IMU-driven approach is taken instead, simply using measurements from the onboard IMU to propagate the states, and correcting the resulting estimates with

encoder feedback and the camera measurements whenever available.

The state estimator is based on the *Unscented Quaternion Estimator* (USQUE) [25]. The USQUE is an *Unscented Kalman Filter* (UKF) [26] which is modified to accomodate the attitude quaternion representation in a manner similar to the modifications made to the *Extended Kalman Filter* (EKF) to obtain a *Multiplicative Extended Kalman Filter* (MEKF) [18]. In [25] the USQUE was found to converge faster than the MEKF when initial attitude estimate errors were larger in the context of satellite attitude estimation. A good guess for the initial attitude of a UAV would be that it is perfectly parallel to the ground. This is in most cases a quite accurate initial guess which begs the question whether there is any real advantage to the USQUE in practice for such a system. Since this particular implementation relies on visually detecting markers, measurements from the camera system may be relatively sparse. This can allow gyro drift to accumulate error in the attitude estimate between the arrival of camera measurements. Since the UAM is equipped with consumer-grade MEMS IMUs this error can be significant even after a relatively short amount of time. Here, the ability of the USQUE to correct the estimate faster can become an advantage.
In this work the USQUE is extended to also provide estimates of the remaining states, where the USQUE was developed specifically for estimating attitude and gyro bias only. This section will start by reviewing the classical UKF, followed by the modifications for the attitude quaternion update as described in [25]. Lastly, the complete state estimator will be introduced.

### 5.2.1 Unscented Kalman Filter

The UKF is a state estimation technique for nonlinear systems of the form:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + \mathbf{w}_k, \tag{5.6a}$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \mathbf{v}_k, \tag{5.6b}$$

where

- $\mathbf{x}_k \in \mathbb{R}^n$ is the state vector,
- $\mathbf{u}_k \in \mathbb{R}^m$ is the vector of control inputs,
- $\mathbf{z}_k \in \mathbb{R}^p$ is the vector of measured outputs,
- $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ and $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$ are vectors of process and measurement noise respectively which are assumed Gaussian,
- $f : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}^n$ and $h : \mathbb{R}^n \mapsto \mathbb{R}^p$ denote the state propagation function and the measurement model respectively.

It can be used as an alternative to the EKF for nonlinear systems where the local linearization of $f$ and $h$ applied by the EKF is not necessarily accurate. The EKF estimates the nonlinear function by a linearization, and since a Gaussian propagated through a linear mapping remains Gaussian, the Gaussian assumption of the Kalman

filter remains valid. The UKF takes a different approach: rather than estimating the nonlinear function, the UKF estimates the distribution after propagating it through the original nonlinear model. It does this by generating a set of points known as sigma points, deterministically sampled from the covariance matrix and propagates those points through the functions $f$ and $h$. It is then simply assumed that the resulting distribution is also Gaussian, and the resulting mean and covariance are computed as a weighted average. This process is known as the *Unscented Transform* (UT).

### The Unscented Transform

There are a few variations of the UT, each with a different procedure for how to compute the weighted average. Only the *scaled UT* as presented in [27] is described here. Furthermore, it is presented in a general form where a mean of a distribution $\hat{\mathbf{x}}$ is given (with no connection to the state in Eq. 5.6), along with its covariance $\mathbf{P_{xx}}$. Initially, the UT must generate the sigma points. With $\hat{\mathbf{x}} \in \mathbb{R}^n$, a total of $2n+1$ sigma points will be generated. The sigma points $\boldsymbol{\chi}_i$ are selected such that the following holds true:

$$\hat{\mathbf{x}} = \sum_{i=0}^{2n} W_i \boldsymbol{\chi}_i, \tag{5.7a}$$

$$\mathbf{P_{xx}} = \sum_{i=0}^{2n} W_i (\boldsymbol{\chi}_i - \hat{\mathbf{x}})(\boldsymbol{\chi}_i - \hat{\mathbf{x}})^T. \tag{5.7b}$$

Which means that the sigma points are each given by:

$$\boldsymbol{\chi}_0 = \hat{\mathbf{x}}, \tag{5.8a}$$

$$\boldsymbol{\chi}_i = \hat{\mathbf{x}} + \sqrt{n+\lambda}\mathbf{c}_i, \qquad i = 1, \dots, n, \tag{5.8b}$$

$$\boldsymbol{\chi}_{n+i} = \hat{\mathbf{x}} - \sqrt{n+\lambda}\mathbf{c}_i, \qquad i = 1, \dots, n, \tag{5.8c}$$

where $\mathbf{c}_i$ is the $i$th column of $\mathbf{C_{xx}}$ which is a matrix square-root typically computed with the Cholesky decomposition of $\mathbf{P_{xx}} = \mathbf{C_{xx}}\mathbf{C_{xx}}^T$, and $\lambda$ is a tuning parameter given by:

$$\lambda = \alpha^2(n + \kappa) - n, \tag{5.9}$$

which can be tuned by varying $\alpha$ and $\kappa$. The weights in Eq. 5.7, $W_i$ are given by:

$$W_0^{[\boldsymbol{\mu}]} = \frac{\lambda}{n+\lambda}, \tag{5.10a}$$

$$W_0^{[\boldsymbol{\Sigma}]} = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta), \tag{5.10b}$$

$$W_i = \frac{1}{2(n+\kappa)}, \qquad i = 1, \dots, 2n, \tag{5.10c}$$

where $W_0^{[\boldsymbol{\mu}]}$ is valid for the mean $\hat{\mathbf{x}}$, and $W_0^{[\boldsymbol{\Sigma}]}$ for the covariance $\mathbf{P_{xx}}$. Here $\beta$ is an extra tuning parameter.

The sigma points $\boldsymbol{\chi}_i$ are propagated through a (nonlinear) function $g(\cdot)$ and the propagated sigma points $\boldsymbol{\xi}_i$ are defined:

$$\boldsymbol{\xi}_i = g(\boldsymbol{\chi}_i). \tag{5.11}$$

Thus the mean and covariance of the transformed points are found by applying the weighted average to the propagated sigma points:

$$\hat{\mathbf{y}} = W_0^{[\boldsymbol{\mu}]}\boldsymbol{\xi}_0 + \sum_{i=1}^{2n} W_i \boldsymbol{\xi}_i, \tag{5.12a}$$

$$\mathbf{P}_{\mathbf{yy}} = W_0^{[\boldsymbol{\Sigma}]}(\boldsymbol{\xi}_0 - \hat{\mathbf{y}})(\boldsymbol{\xi}_0 - \hat{\mathbf{y}})^T + \sum_{i=1}^{2n} W_i (\boldsymbol{\xi}_i - \hat{\mathbf{y}})(\boldsymbol{\xi}_i - \hat{\mathbf{y}})^T, \tag{5.12b}$$

where $\hat{\mathbf{y}}$ is the output mean, i.e. the mean of the transformed distribution and $\mathbf{P}_{\mathbf{yy}}$ the output covariance. This procedure is used in a Kalman filter-like structure to estimate the time propagation of the state and the expectation of the measurement model given the state estimate. [27]

### UKF Equations

Recall the discrete-time nonlinear state space model described in Eq. 5.6. The UT is used to obtain the estimates $\hat{\mathbf{x}}_{k+1}^-$ and $\hat{\mathbf{z}}_k$, where the $\hat{\mathbf{x}}^-$ denotes a time-propagated estimate (also often described as the prediction step) and $\hat{\mathbf{x}}^+$ denotes a state estimate after correction, using measurements. In a linear or an Extended Kalman filter [27], the Kalman gain is usually given by:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}. \tag{5.13}$$

The matrix $\mathbf{P}_k^-$ is given by $\mathbf{P}_{\mathbf{xx}}$ of the UT during a time-update, where $\hat{\mathbf{x}}_k^-$ is obtained by applying the UT to $\hat{\mathbf{x}}_{k-1}^+$ with $f$ as the propagation function. To accommodate the process noise, its covariance is added:

$$\mathbf{P}_k^- = \mathbf{P}_{\mathbf{xx}} + \mathbf{Q}_k. \tag{5.14}$$

The matrix $\mathbf{H}_k$, assumes a linear measurement model, which is not generally the case here. Instead, the term $\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T$ is obtained from the UT of $\hat{\mathbf{x}}_k^-$ with the nonlinear function $h$ as the propagation function, where $\hat{\mathbf{z}}_k$ is estimated:

$$\mathbf{P}_{\mathbf{zz}} = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T. \tag{5.15}$$

The term $\mathbf{P}_k^- \mathbf{H}_k^T$ corresponds to the cross-correlation matrix between $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}$ which can be approximated as:

$$\mathbf{P}_k^- \mathbf{H}_k^T \approx \mathbf{P}_{\mathbf{xz}} = \sum_{i=0}^{2n} W_i (\boldsymbol{\chi}_i - \hat{\mathbf{x}}_k^-)(\boldsymbol{\xi}_i - \hat{\mathbf{z}}_k)^T, \tag{5.16}$$

which leads to the Kalman gain being computed as:

$$\mathbf{K}_k = \mathbf{P}_{\mathbf{xz}} (\mathbf{P}_{\mathbf{zz}} + \mathbf{R}_k)^{-1}. \tag{5.17}$$

The update equations for the UKF are then given by:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k), \tag{5.18a}$$
$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k {\mathbf{P}_{\mathbf{xz}}}^T. \tag{5.18b}$$

The full set of equations for the UKF can thus be summarized as:

**Time update**

$$\hat{\mathbf{x}}_k^- = UT\left(\hat{\mathbf{x}}_{k-1}^+, \mathbf{C}_{k-1}^+, f(\cdot)\right), \tag{5.19a}$$
$$\mathbf{P}_k^- = \mathbf{P}_{\mathbf{xx}} + \mathbf{Q}_k, \tag{5.19b}$$

**Measurement update**

$$\hat{\mathbf{z}}_k = UT\left(\hat{\mathbf{x}}_k^-, \mathbf{C}_k^-, h(\cdot)\right), \tag{5.19c}$$
$$\mathbf{P}_{\mathbf{xz}} = \sum_{i=0}^{2n} W_i(\boldsymbol{\chi}_i - \hat{\mathbf{x}}_k^-)(\boldsymbol{\xi}_i - \hat{\mathbf{z}}_k)^T, \tag{5.19d}$$
$$\mathbf{K}_k = \mathbf{P}_{\mathbf{xz}}\left(\mathbf{P}_{\mathbf{zz}} + \mathbf{R}_{\mathbf{k}}\right)^{-1}, \tag{5.19e}$$
$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - \hat{\mathbf{z}}_k), \tag{5.19f}$$
$$\mathbf{P}_k^+ = \mathbf{P}_k^- - \mathbf{K}_k {\mathbf{P}_{\mathbf{xz}}}^T. \tag{5.19g}$$

### 5.2.2 Unscented Quaternion Estimator

The reason that is necessary to modify the approach outlined in Eqs. 5.19a-5.19g in order to accomodate for quaternion representation of the attitude, is that the group of unit quaternions is not closed under addition. The kinematics of the quaternion are multiplicative (see Eg. 3.16). When computing the weighted average of the UT and when performing the measurement update of the state, it is clear that the attitude quaternion will be subjected to many additions if this filter is directly applied. This may result in the norm of $\hat{\mathbf{q}}$ deviating from 1. Under the assumption that the error is small, this can be mitigated by simply normalizing the quaternion. However, this does deteriorate the accuracy of the attitude estimate. Instead, the procedure may be modified by a change of variable, and some well placed quaternion multiplications.
The general idea is to create an error state for the quaternion, which represents the change in state between the current time instance and the previous time instance (in a discrete-time setting). The error state for a unit quaternion is thus a small rotation between the two time steps such that the quaternion $\mathbf{q}_k$ at time step $k$ is given by:

$$\mathbf{q}_k = \mathbf{q}_{k-1} \odot \delta\mathbf{q}_k, \tag{5.20}$$

where $\mathbf{q}_{k-1}$ is the attitude quaternion representing the attitude at time $k-1$ and $\delta\mathbf{q}_k$ is the error quaternion representing the rotation between the two. Assuming that the sampling period for the discrete-time system is relatively short compared to

the angular velocity of the system, then the rotation represented by $\delta\mathbf{q}_k$ is relatively small. Thus we can change to a different representation of attitude $\delta\boldsymbol{\vartheta}$ which is not restricted to multiplications only. It is chosen such that $\|\delta\boldsymbol{\vartheta}\| \approx \theta$ for small rotations where $\theta$ is the angle given in Eq. 3.6 for the error quaternion [28]. Typical representations are the *Rodrigues Parameters* (RP, also known as the *Gibbs vector*) or the *Modified Rodrigues Parameters* (MRP). These are typically chosen because they are minimal representations (i.e. they have 3 components instead of 4 like the quaternions), and they linearize to half angles and quarter angles respectively. In [25] these are generalized, to a representation with 2 parameters $a$ and $f_p$ where $f_p = 2(a+1)$. Then $\delta\boldsymbol{\vartheta}$ is given by:

$$\delta\boldsymbol{\vartheta} = \frac{f_p}{a + \delta q_0}\delta\mathbf{q}_{1:3}, \tag{5.21}$$

where the RPs corresponds to $a = 0$ and $f_p = 2$, and the MRPs correspond to $a = 1$ and $f_p = 4$. The mapping from the 3 component error to the error quaternion is given by:

$$\delta q_0 = \frac{-a\|\delta\boldsymbol{\vartheta}\|^2 + f_p\sqrt{f_p^2 + (1-a^2)\|\delta\boldsymbol{\vartheta}\|^2}}{f_p^2 + \|\delta\boldsymbol{\vartheta}\|^2}, \tag{5.22a}$$

$$\delta\mathbf{q}_{1:3} = \frac{a + \delta q_0}{f_p}\delta\boldsymbol{\vartheta}. \tag{5.22b}$$

The procedure is then to define a state vector $\hat{\mathbf{x}}$ which is separated:

$$\hat{\mathbf{x}} = \begin{bmatrix} \delta\hat{\boldsymbol{\vartheta}}_k \\ \hat{\boldsymbol{\beta}}_k \end{bmatrix}, \tag{5.23}$$

where $\hat{\mathbf{a}}_k$ is the estimated 3-component attitude error, and $\hat{\boldsymbol{\beta}}_k$ the estimated gyro bias. The state propagation and measurement functions must be written as functions of $\hat{\mathbf{a}}_k$ instead of the attitude quaternion since it is this attitude error state that will be propagated and updated as described in Eq. 5.19. However, the quaternion kinematics of Eq. 3.16 are functions of the quaternion itself. Therefore, mapping back and forth between $\mathbf{q}$ and $\delta\boldsymbol{\vartheta}$ is done instead.
The USQUE is initialized with an initial attitude quaternion $\hat{\mathbf{q}}_0^+$ and initial state estimate:

$$\hat{\mathbf{x}}_0^+ = \begin{bmatrix} \mathbf{0} \\ \hat{\boldsymbol{\beta}}_0^+ \end{bmatrix}, \tag{5.24}$$

and an initial corresponding covariance matrix $\mathbf{P}_0^+$. The sigma points are calculated using Eq. 5.8, and for each sigma point $\boldsymbol{\chi}_i$ the corresponding error quaternion $\delta\mathbf{q}_i$ is calculated with Eq. 5.22. This is immediately followed by computing the full attitude quaternion for each sigma point by applying Eq. 5.20 to each sigma point:

$$\hat{\mathbf{q}}_{k-1,i}^+ = \hat{\mathbf{q}}_{k-1}^+ \odot \delta\hat{\mathbf{q}}_{k-1,i} \quad \forall \quad i = 0, \ldots, n, \tag{5.25}$$

where $\hat{\mathbf{q}}_{k-1}^+$ is the most recent estimate of the attitude quaternion. These quaternions are propagated forward in time using Eq. 3.16 where the angular velocity $\hat{\boldsymbol{\omega}}_k$ is the gyro measurement with the bias estimate subtracted.

$$\hat{\mathbf{q}}_{k,i}^- = \Omega(\hat{\boldsymbol{\omega}}_k)\hat{\mathbf{q}}_{k-1,i}^+ \tag{5.26}$$

In order to obtain the state estimates the quaternion from each sigma point is then converted back to a 3 component error representation, first by computing:

$$\delta\hat{\mathbf{q}}_{k,i}^{-} = \left(\hat{\mathbf{q}}_{k,0}^{-}\right)^{-1} \odot \hat{\mathbf{q}}_{k,i}^{-}, \tag{5.27}$$

and then by applying Eq. 5.21. Then the estimates of the mean and covariance can be computed using Eq. 5.12 as usual in the UT. [25]
For the measurement update, the attitude measurements arrive as quaternion measurements, as described in section 5.1. Therefore, the measurement goes through the same procedure of mapping to the 3 component attitude error representation:

$$\delta\mathbf{q}_m = \left(\hat{\mathbf{q}}_{k,0}^{-} \odot \delta\mathbf{q}(\hat{\mathbf{a}}_k^{-})\right)^{-1} \odot \mathbf{q}_m, \tag{5.28}$$

$$\delta\boldsymbol{\vartheta}_m = \frac{f_p}{a + \delta q_{m_0}} \delta\mathbf{q}_{m_{1:3}}. \tag{5.29}$$

At this point, it is possible to apply the update equations (Eqs. 5.19e-5.19g) of the UKF as normal, and when this is completed the corrected attitude quaternion can be computed by once again applying Eqs. 5.22 and 5.20.

$$\hat{\mathbf{q}}_k^{+} = \hat{\mathbf{q}}_{k,0}^{-} \odot \delta\hat{\mathbf{q}}(\delta\boldsymbol{\vartheta}_k^{+}) \tag{5.30}$$

To finish up, the attitude error state $\delta\hat{\boldsymbol{\vartheta}}_k^{+}$ must be reset to $\mathbf{0}$.

### 5.2.3 Unscented Estimator for a UAM

In order for the state estimator to be used for the UAM, the state space is extended to include the remaining necessary states. The full state vector for this estimator is written as:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{p}}_B \\ \hat{\mathbf{q}} \\ \hat{\boldsymbol{\theta}} \\ \hat{\boldsymbol{\nu}} \\ \hat{\boldsymbol{\varpi}} \\ \hat{\boldsymbol{\beta}}_a \\ \hat{\boldsymbol{\beta}}_g \end{bmatrix} \in \mathbb{R}^{16+2N_j}, \tag{5.31}$$

where the notation $(\hat{\cdot})$ denotes that the quantity is an estimate and

- $\hat{\mathbf{p}}_B \in \mathbb{R}^3$ is the position of the quadrotor body frame relative to the inertial frame,
- $\hat{\mathbf{q}} \in \mathbb{R}^4$ is the vector of coefficients of the attitude quaternion,
- $\hat{\boldsymbol{\theta}} \in \mathbb{R}^{N_j}$ is the vector of manipulator joint positions,
- $\hat{\boldsymbol{\nu}} \in \mathbb{R}^3$ is the velocity of the quadrotor body frame relative to the inertial frame,
- $\hat{\boldsymbol{\varpi}} \in \mathbb{R}^{N_j}$ is the manipulator joint velocities,

- $\hat{\boldsymbol{\beta}}_a \in \mathbb{R}^3$ is the accelerometer bias,
- $\hat{\boldsymbol{\beta}}_g \in \mathbb{R}^3$ is the gyro bias.

The filter is implemented with an IMU-driven approach, that is, inputs to the filter are the bias-corrected gyro and accelerometer measurements. The camera measurements along with encoder feedback is used for the correction in the filter equations. Therefore, the vector of inputs is given by:

$$\mathbf{u} = \begin{bmatrix} \hat{\boldsymbol{a}} \\ \hat{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_m - \boldsymbol{\beta}_a \\ \boldsymbol{\omega}_m - \boldsymbol{\beta}_g \end{bmatrix} \in \mathbb{R}^6, \tag{5.32}$$

where

- $\hat{\boldsymbol{a}} \in \mathbb{R}^3$ is the bias-corrected accelerometer measurement,
- $\hat{\boldsymbol{\omega}} \in \mathbb{R}^3$ is the bias-corrected gyro measurement.

The vector of measured outputs of this state-space system is:

$$\mathbf{z} = \begin{bmatrix} \mathbf{p}_m \\ \mathbf{q}_m \\ \boldsymbol{\theta}_m \\ \boldsymbol{\varpi}_m \end{bmatrix} \in \mathbb{R}^{13}, \tag{5.33}$$

where

- $\mathbf{p}_m \in \mathbb{R}^3$ is the measured position of the quadrotor body in the inertial frame,
- $\mathbf{q}_m \in \mathbb{R}^4$ is the measured attitude of the quadrotor body,
- $\boldsymbol{\theta}_m$ is the measured manipulator joint positions,
- $\boldsymbol{\varpi}_m$ is the measured manipulator joint velocities.

With an approach similar to the one in [29] the discrete-time difference equation for the state (using Forward-Euler discretization of $\dot{\mathbf{x}}$) is then given as:

$$\mathbf{p}_{B,k+1} = \mathbf{p}_{B,k} + \boldsymbol{\nu}\Delta t, \tag{5.34a}$$
$$\mathbf{q}_{k+1} = \Omega\left(\hat{\boldsymbol{\omega}}\right)\mathbf{q}_k, \tag{5.34b}$$
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{\varpi}_k\Delta t, \tag{5.34c}$$
$$\boldsymbol{\nu}_{k+1} = \boldsymbol{\nu}_k + \left({}^{\mathcal{W}}_{B}R(\mathbf{q}_k)\,\hat{\boldsymbol{a}}_k + \mathbf{g}\right)\Delta t, \tag{5.34d}$$
$$\boldsymbol{\varpi}_{k+1} = \boldsymbol{\varpi}_k, \tag{5.34e}$$
$$\boldsymbol{\beta}_{a,k+1} = \boldsymbol{\beta}_{a,k}, \tag{5.34f}$$
$$\boldsymbol{\beta}_{g,k+1} = \boldsymbol{\beta}_{g,k}, \tag{5.34g}$$

where $\mathbf{g} = \begin{bmatrix} 0 & 0 & g \end{bmatrix}^T$ is the vector of gravitational acceleration expressed in the inertial frame, and ${}^{\mathcal{W}}_{B}R(\mathbf{q}_k)$ is the rotation between the body frame and the inertial frame. Since the quaternion requires special treatment in the estimator, as explained in the previous section, another state vector $\delta\mathbf{x}$ and measurement model $\delta\mathbf{z}$ is defined,

which is used internally in the estimator. Here the quaternion itself is replaced by the attitude error component and the two are given by:

$$\delta\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{p}}_B \\ \delta\boldsymbol{\vartheta} \\ \hat{\boldsymbol{\theta}} \\ \hat{\boldsymbol{\nu}} \\ \hat{\boldsymbol{\varpi}} \\ \hat{\boldsymbol{\beta}}_a \\ \hat{\boldsymbol{\beta}}_g \end{bmatrix} \in \mathbb{R}^{15+2N_j}, \tag{5.35}$$

$$\delta\mathbf{z} = \begin{bmatrix} \mathbf{p}_m \\ \delta\boldsymbol{\vartheta}_m \\ \boldsymbol{\theta}_m \\ \boldsymbol{\varpi}_m \end{bmatrix} \in \mathbb{R}^{12}, \tag{5.36}$$

where $\delta\boldsymbol{\vartheta}$ is the attitude error and $\delta\boldsymbol{\vartheta}_m$ is the measured attitude error which was previously described by Eq. 5.29. This state vector is used in the unscented filter. The state propagation function $f\left(\delta\mathbf{x}_k, \mathbf{u}_k\right)$ is given by Eqs. 5.34 except for the attitude error difference which requires the mappings of the USQUE to be applied. As explained in section 5.2.2, this procedure ultimately applies the quaternion kinematics of Eq. 5.34b. The observation model $h(\delta\mathbf{x}_k) : \mathbb{R}^{15+2N_j} \mapsto \mathbb{R}^{12}$ is given by:

$$h(\delta\mathbf{x}_k) = \begin{bmatrix} \mathbf{p}_{B,k} \\ \delta\boldsymbol{\vartheta}_k \\ \boldsymbol{\theta}_k \\ \boldsymbol{\varpi}_k \end{bmatrix}. \tag{5.37}$$

With this setup, the UKF equations (Eqs. 5.19a-5.19g) can be applied as usual. It is worth mentioning that the manipulator joint position state can be excluded from the state estimator, instead relying on measurements directly with no significant impact on accuracy, due to the high quality of the encoder measurements. Here it is included simply for the sake of completeness.

# Part III

# Experiments & Results

# 6 | Simulations & Experiments

This chapter includes simulations of the dynamic models and a test verifying the performance of the state estimator. The results will be stated along with a comment about what was expected. The results will be discussed in further detail in chapter 7.

The first dynamic modelling attempt was to use the RNE algorithm, described in test 6.1, and the presented simulation serves to illustrate an issue with the model solution. The second attempt at modelling the dynamics was to use the Euler-Lagrange method, described in tests 6.2, 6.3 and 6.4. These simulations are focused on the effects of the manipulator torque and manipulator motion on the quadrotor body, as this is the key component of the coupled dynamics of the system. The simulated cases are fairly simple as the coupled dynamics can make the results very complicated to interpret if multiple effects are simulated simultaneously.

## 6.1 Newton-Euler Simulation

### Description

To assess the dynamic model, an initial position of $-\pi/2$ rad was set to the second joint. Moreover, no torque was applied to any of the joint actuators. The thrust force of the rotors negates the force of gravity, and it is equal at each of the four rotors. In this particular configuration the link is expected to swing down as a result of the gravitational forces. The UAM is expected to move a little in the positive direction of the inertial frame $x$-axis, as a result of the forces caused by the configuration and motion of the manipulator.

### Result

According to Fig. 6.1 both the second link and the UAV remain in their original positions. This result is clearly incorrect and points to an error in the solution of the coupled dynamic equations.

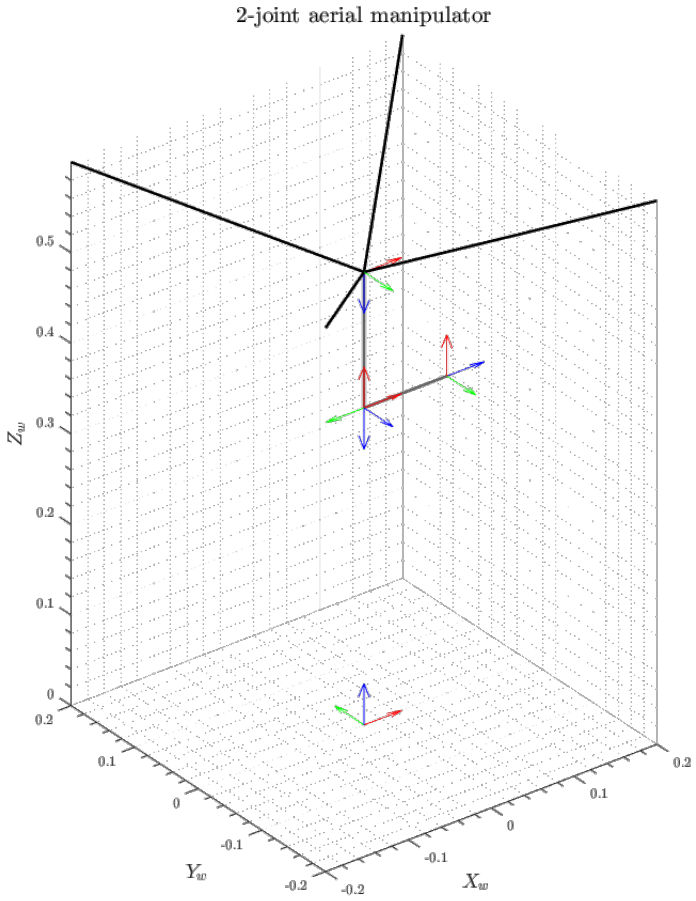



**Figure 6.1:** Newton-Euler simulation performance given the joint 2 initial angle of $\theta_2 = -\pi/2$ rad

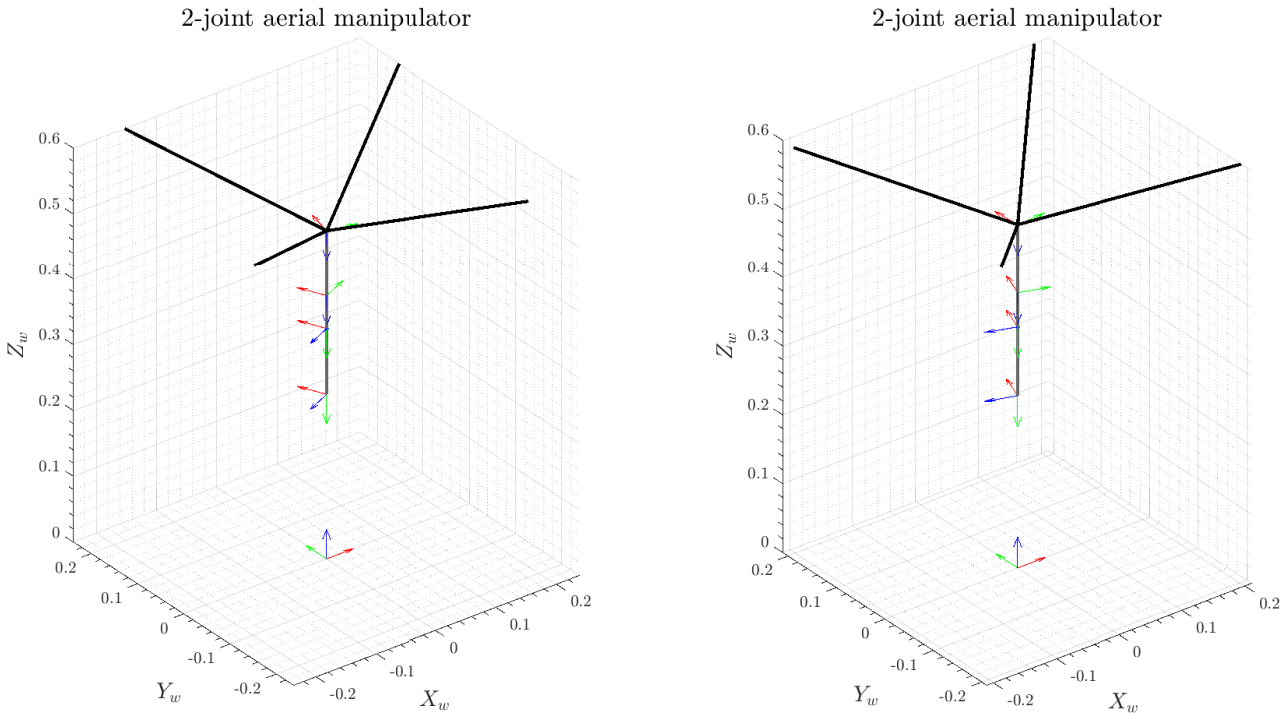
## 6.2 Simulation: Joint 1 Torque

### Description

The initial setup consists of a constant torque of 0.1 Nm applied to the first joint servo motor. The second joint is static. The initial positions for both actuators are zero radians, i.e. it is freely hanging downwards. The thrust force of the rotors negates the force of gravity, and it is equal at each of the four rotors. The expected behaviour given this configuration would be to have the manipulator rotate in one direction, and the quadrotor rotate in the opposite direction. This is because the manipulator and the quadrotor interact at the same point at the base link, and according to the Newton's 3rd law, every action (force) has an equal but opposite reaction. Thus, in this case the manipulator exerts a force on the quadrotor, which results in the quadrotor also exerting an equal but opposite force onto the manipulator. Moreover, given the forces applied at the rotors, the quadrotor is expected to keep its location constant and equal to its initial location.

## Result

According to Fig. 6.2, the expected rotational behaviour mentioned in the description of this simulation is fulfilled. Fig. 6.3 describes the position, velocity and acceleration which are all constant. The quadrotor's initial position is 0.5 meters above ground and it stays that way during the entire simulation time. On top of that, the quaternion, and its time derivatives are changing over time. The quaternion can be difficult to interpret, but it is noticeable that only the first two of the coefficients in the vector part are changing, indicating that it is rotating in the $xy$-plane. This is expected as it describes the rotational movement reaction of the quadrotor. Finally, in Fig. 6.4 the manipulator joint position, velocity, and acceleration is displayed. The joint velocity is observed to peak after around 4s before it starts dropping off. This may come as a surprise with a constant torque applied, but it is due to the rotation of the quadrotor, which also causes the manipulator to rotate, that the joint torque must also counteract. Eventually the two rotations must fall into an equilibrium that happens when the joint acceleration stabilizes at 0. The reason for this "overshoot-like" behaviour where the joint reaches a higher velocity than its steady state is that the manipulator has a significantly smaller moment of inertia than the quadrotor, allowing it to build up velocity faster than the quadrotor. Once the quadrotor starts building velocity, the joint rotation is then slowed down.




**(a)** Configuration at time $t_1 < t_2$

**(b)** Configuration at time $t_2$

**Figure 6.2:** Simulation: A torque of 0.1 Nm is applied on the joint 1 servo motor. The initial joint positions are zero radians. Notice the manipulator frames have negative rotation about the inertial frame $z$-axis, while the quadrotor frame has positive rotation.

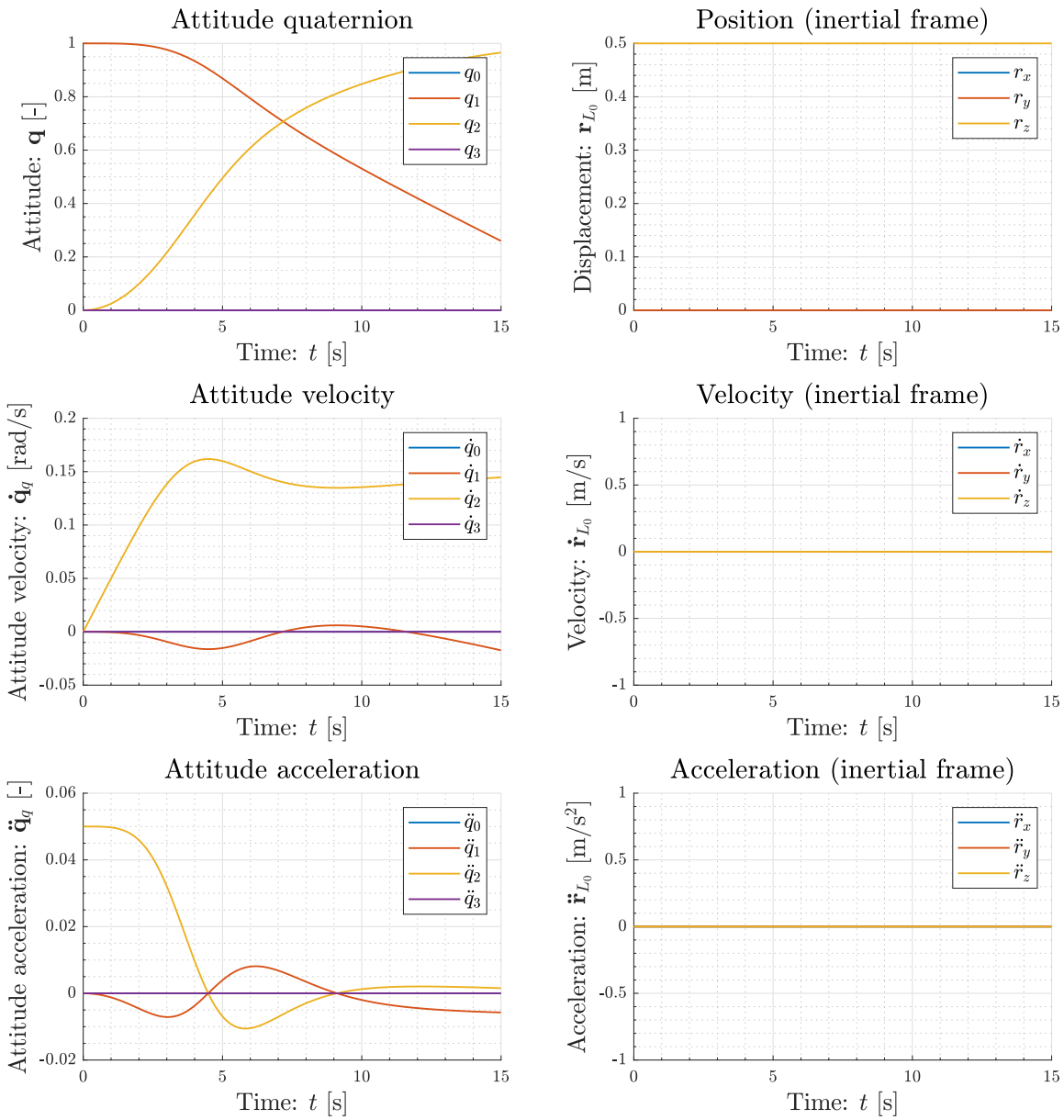



**Figure 6.3:** Quadrotor generalized coordinates and the first and second derivatives throughout the simulation.

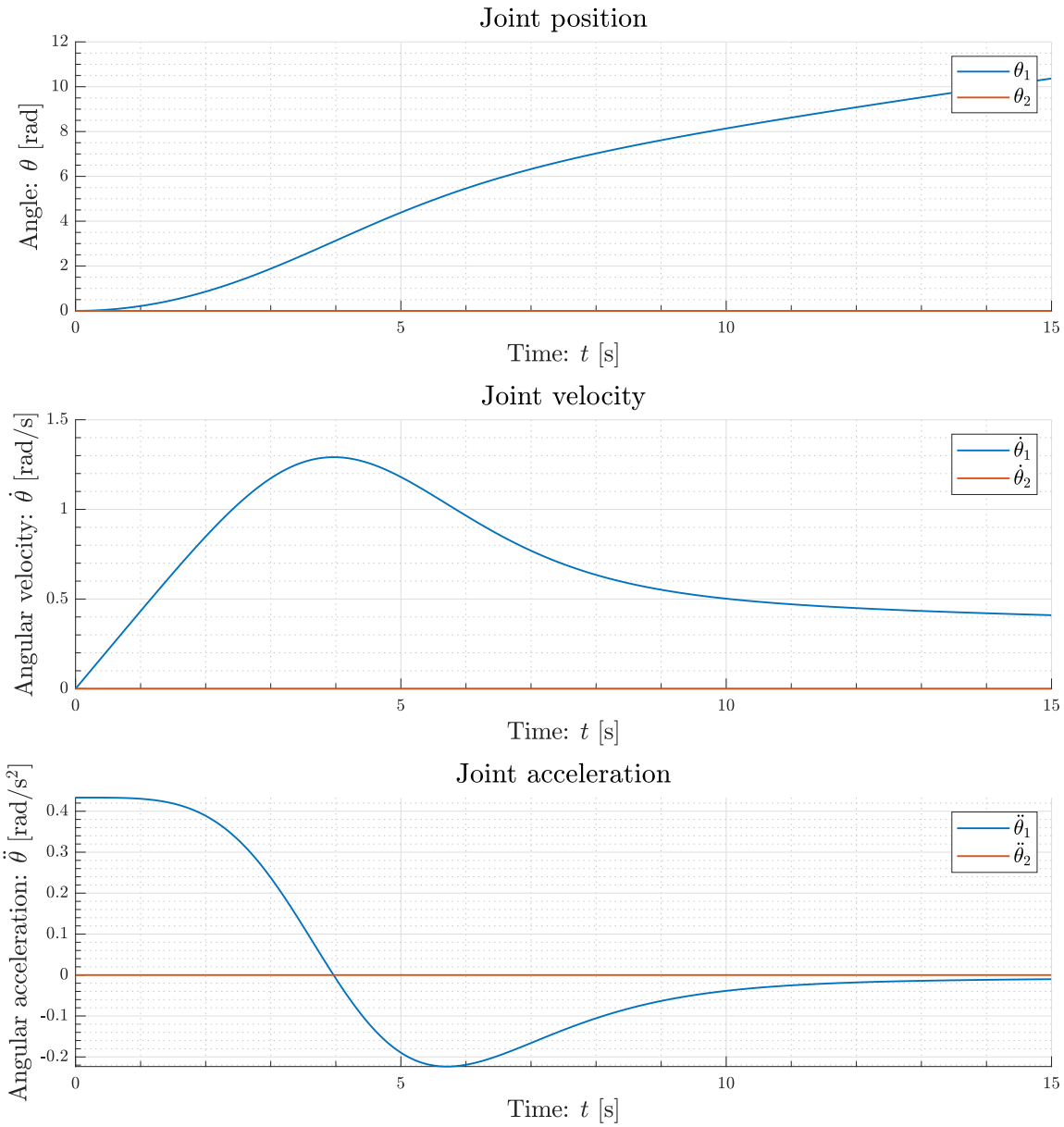



**Figure 6.4:** Plot of manipulator joint variables over time for the simulation a with constant joint 1 torque applied.

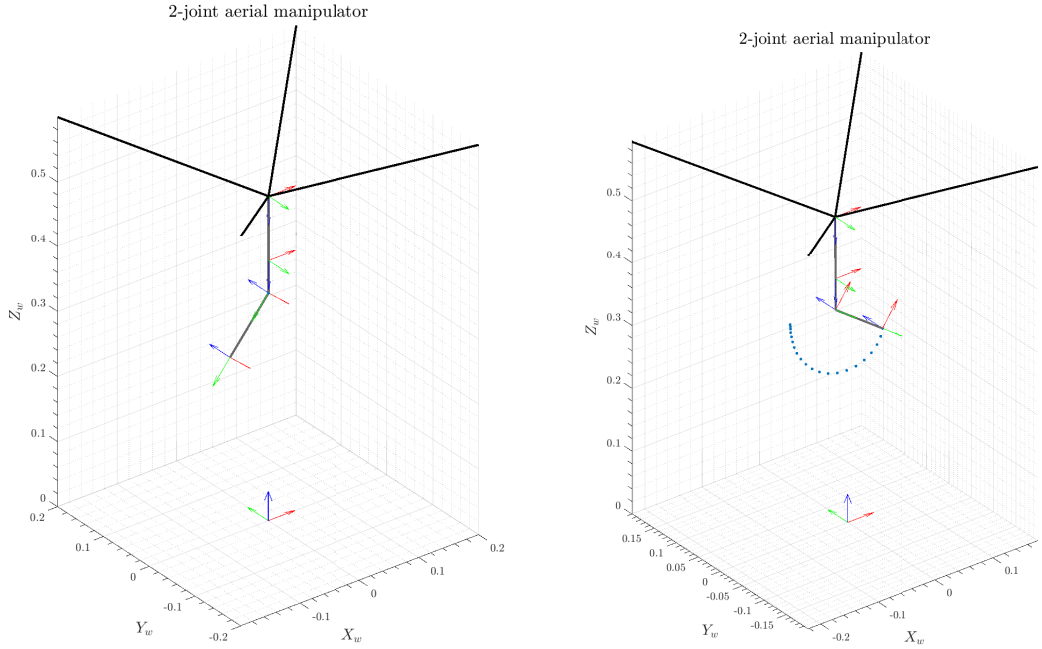
## 6.3 Simulation: Joint 2 Angle

### Description

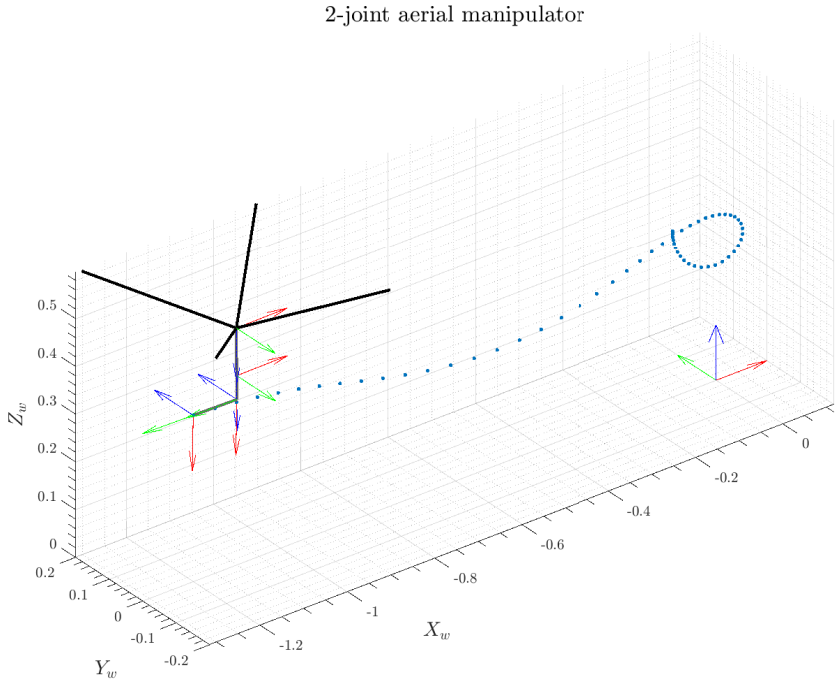
The setup in this test is similar to the one presented in test 6.1, with the only difference being the initial position of the second joint being equal to $\pi/2$ instead of $-\pi/2$. As a result of this, the drifting direction of the quadrotor should be negative along the inertial frame $x$-axis. The choice of the initial angle, whether it is positive or negative is arbitrary. No joint torques are applied and the rotor rotation speeds (equal at all rotors) is set such that it nullifies the force of gravity. To make the effect of the manipulator's initial position more visible the mass and moment of inertia of the manipulator have been scaled up compared to the physical system.

### Result

According to Figs. 6.5a and 6.5b the manipulator is swinging as expected, shifting the joint angle slightly as the quadrotor starts tilting which can be observed when comparing the peaks in the plot. This is also displayed in Fig. 6.7. Moreover, as can be seen in Fig. 6.5c the quadrotor's simulated motion corresponds to the expected physical response given a force caused by the manipulator's response to gravity. The path is marked by the blue dots. Fig. 6.6 showcases the states (and their time derivatives) of the quadrotor. The position of its CoM is changing over time. It moves in the negative direction of the $x$-axis, and it slightly wobbles on the $z$-axis, but as the simulation progresses and the quadrotor tilts further it starts dropping as expected. The wobbling is caused by the motion of the manipulator which in turn also causes the tilt. As a consequence, the velocity and acceleration on the $x$- and $z$-axes are also growing in the negative direction.




**(a)** **(b)**




**(c)**

**Figure 6.5:** Simulation: **(a)** Joint 2 set at an initial position of $\pi/2$, here shortly after starting the simulation. **(b)** Link 2 swings down due to gravitation. **(c)** The UAM base starts moving due to the effects of the shifting center of gravity. The blue dots indicate the trajectory of the end-effector.

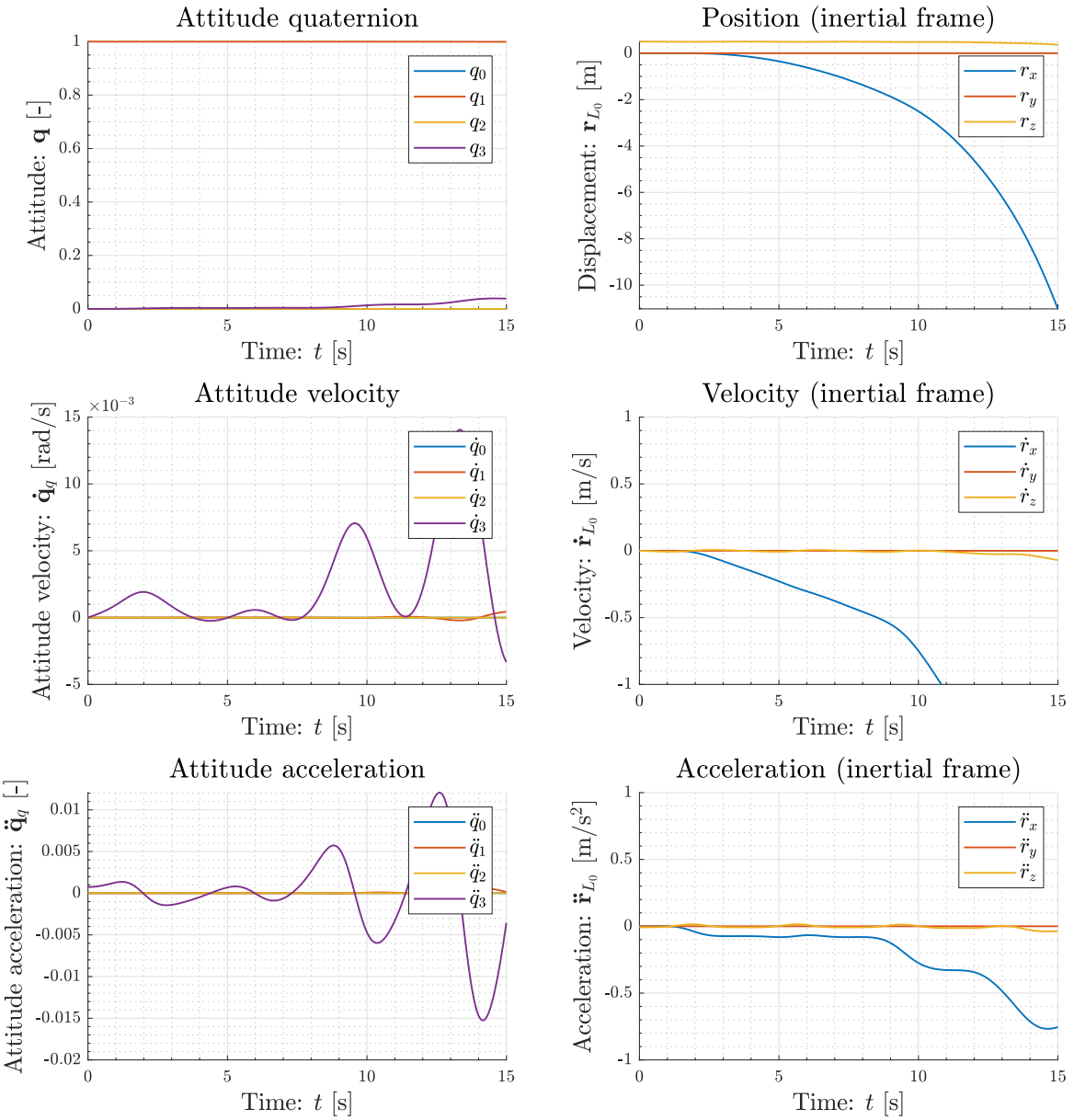



**Figure 6.6:** Quadrotor generalized coordinates and the first and second derivatives throughout the simulation without joint torques and initial condition $\theta_2 = \pi/2$. The quaternion reveals the gradually increasing tilt of the quadrotor body.

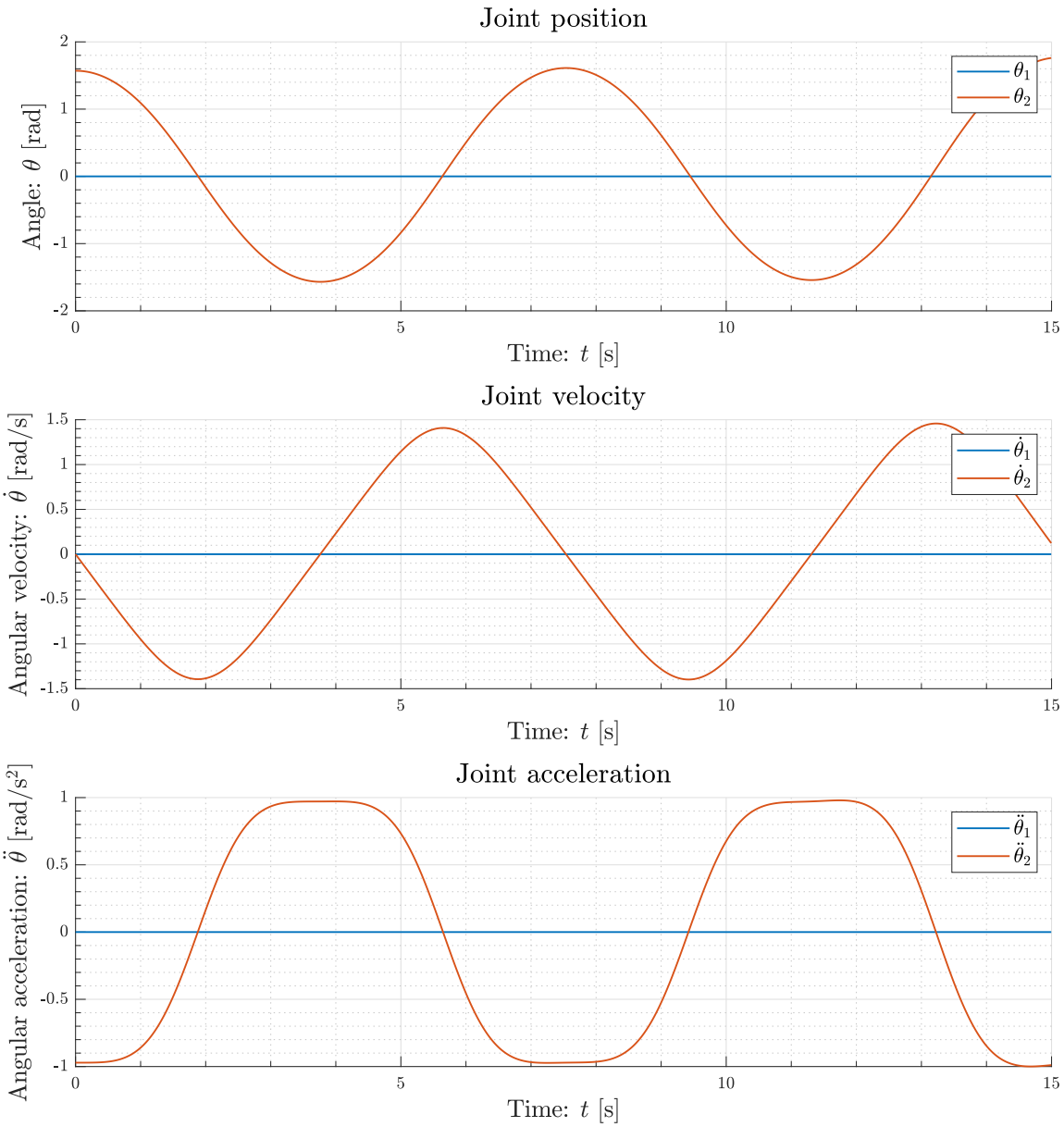



**Figure 6.7:** Plot of manipulator joint variables over time for the simulation a without joint torques. Observing the peaks of the joint position plot reveals that the angles shift as the quadrotor begins tilting.

## 6.4 Simulation: Joint 2 Torque

### Description

This simulation investigates the effect on the quadrotor body of torques applied to the second joint. Both manipulator joints are initialized at their zero positions. The thrust force is again set to nullify gravity when the quadrotor is not tilted, and now a constant torque of 0.05 Nm is applied by the second joint servo. The second link of the manipulator is expected to rotate up, and the reaction force (torque) is expected to cause the quadrotor to tilt down in the opposite direction. This should cause the thrust force to get angled, making the system move in the direction the two bodies rotate towards, also causing it to lose altitude due to the loss of upwards thrust. The mass and moment of inertia of the manipulator are scaled up to make the effects more clear.

### Result

In Fig. 6.8, the result of the applied torque makes it clear that the manipulator link and the quadrotor body rotate towards each other as expected. It can also be seen that the UAM moves along the inertial frame $x$-axis in the negative direction as expected. The drop in altitude is just barely visible in Fig. 6.9 due to the short simulation time, but the downwards acceleration is easy to see. Fig. 6.10 displays the joint variables, where the second joint acts as expected, slowing down as it moves up, mainly due to increased torque caused by the force of gravity acting on the CoM of link 2.

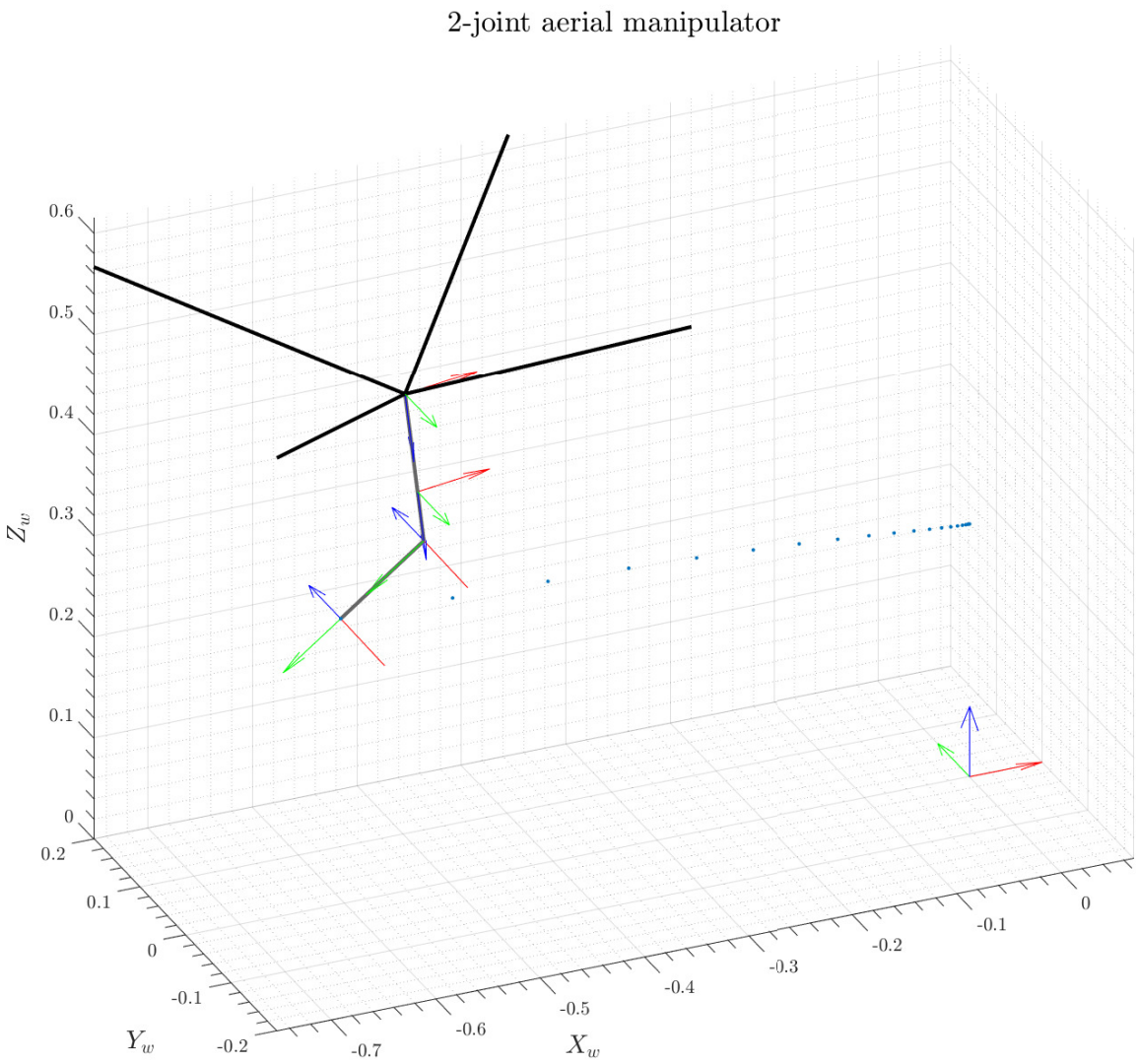



**Figure 6.8:** Simulation: A torque of 0.05 Nm is applied on the joint 2 servo motor. The initial joint positions are zero radians.

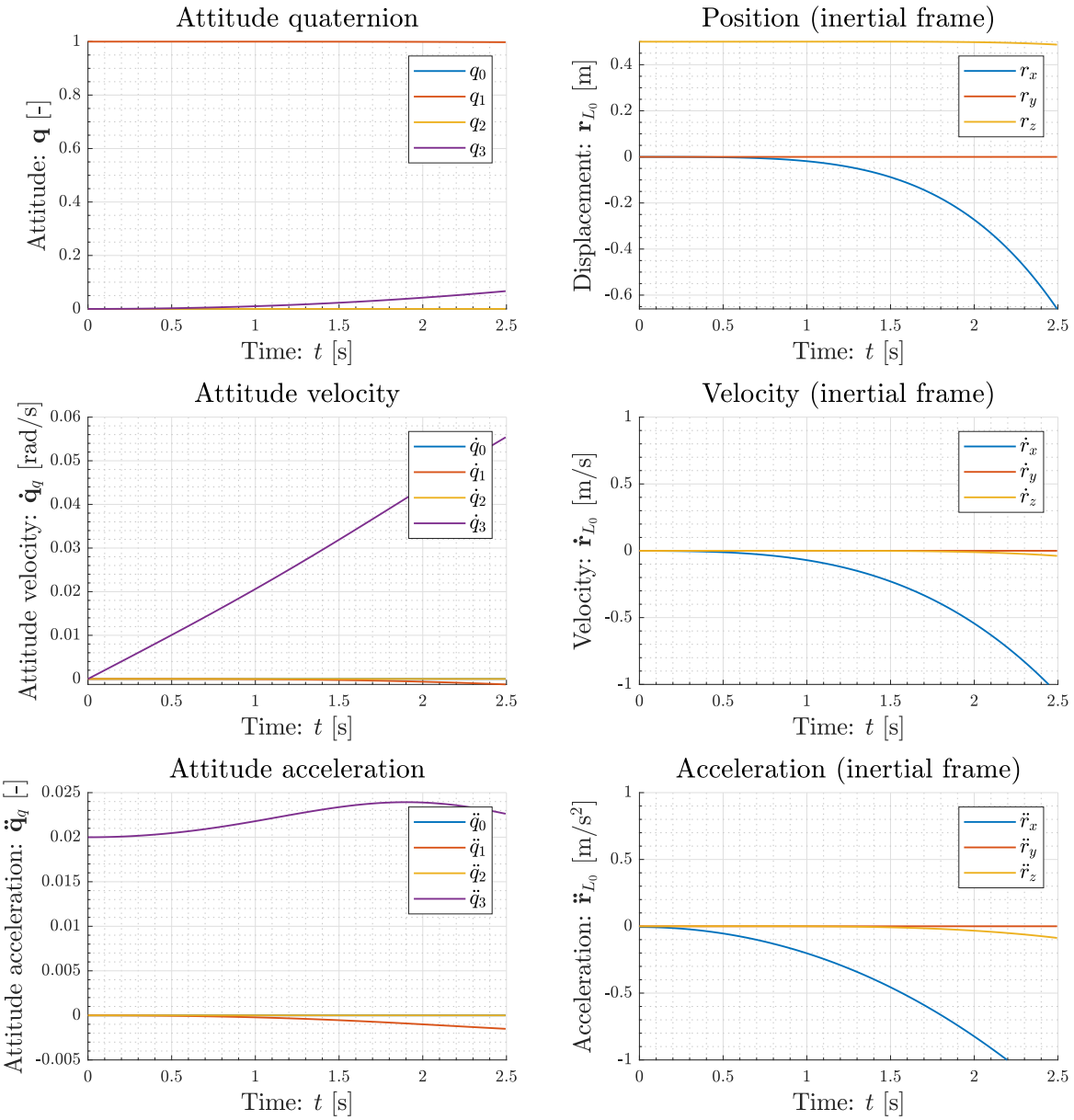



**Figure 6.9:** Quadrotor generalized coordinates and the first and second derivatives throughout the simulation.

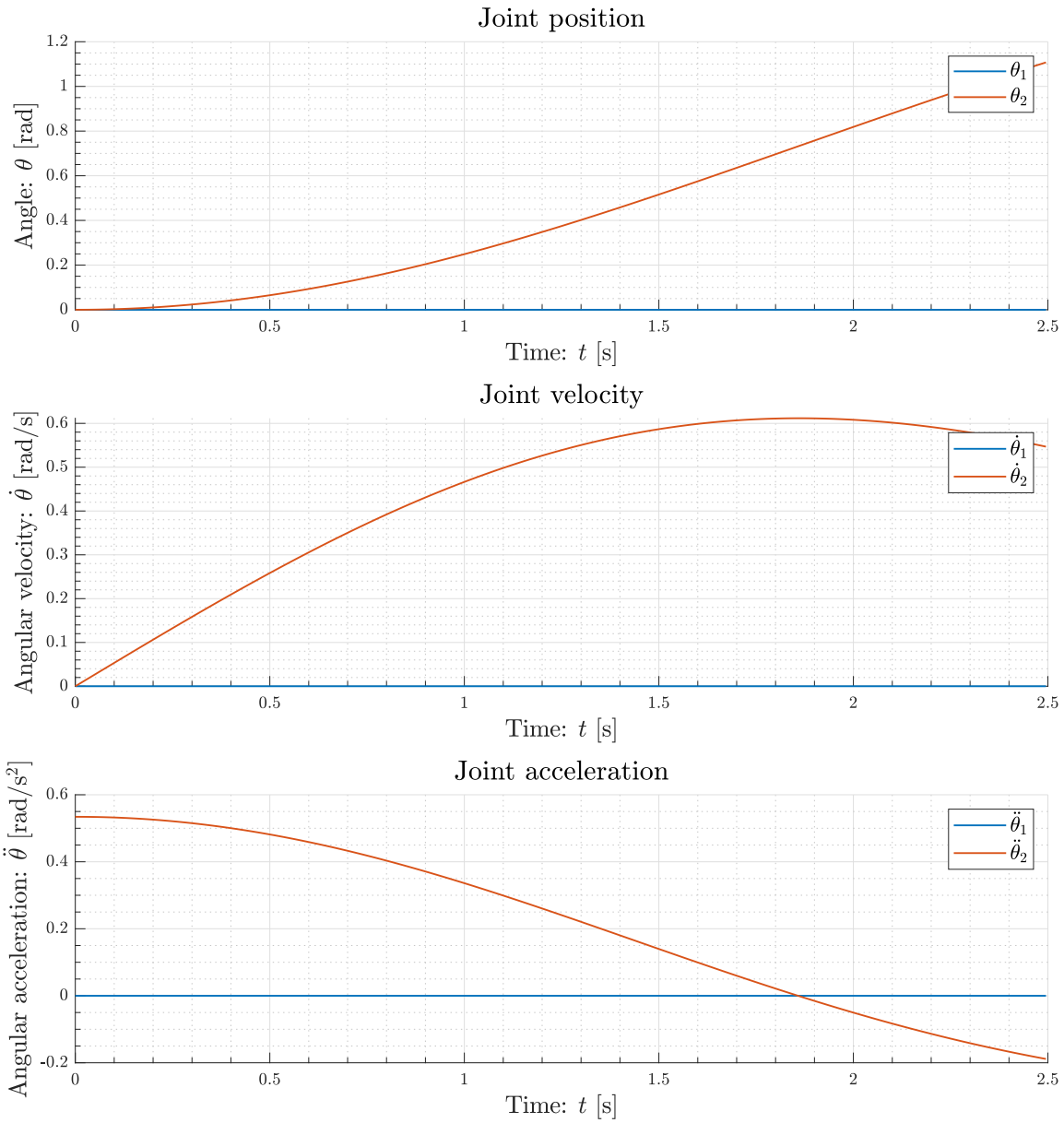



**Figure 6.10:** Test 3: Representation of manipulator joint states over time

## 6.5 State Estimator Validation

### Description

This test is performed by collecting data with the IMU and the camera. The sensors start the test at rest. They are then arbitrarily moved around the room while the camera is facing an ArUco marker, which it should use to estimate its pose. It is expected that the estimates should quickly drift while camera measurements are not available, but should be corrected at measurement updates.

### Results

In Fig. 6.11 the attitude estimate is displayed. Here, there are some noticeable very large steps in the attitude quaternion. Analysis of the raw sensor data reveals that this is caused by the $z$-axis of the ArUco marker occasionally being flipped. It is not known for certain why this occurs, but it is likely that it is due to motion blur. The particular camera available appears to be sensitive to motion, which is exacerbated by the fact that camera measurements arrive very slowly, providing only few frames per second with most not being useful for detecting markers due to the motion blur. Due to the lack of camera measurements, the accelerometer is simply continuously integrated without correction which causes the errors to accumulate quickly. In Fig. 6.12 the estimated positions are shown. Here, the camera measurements are deliberately left out, as they cause similar steps in the attitude estimates when included. Leaving out camera measurements from the attitude estimates also provides results that are more reasonable than when camera measurements are included, as seen in Fig. 6.13.

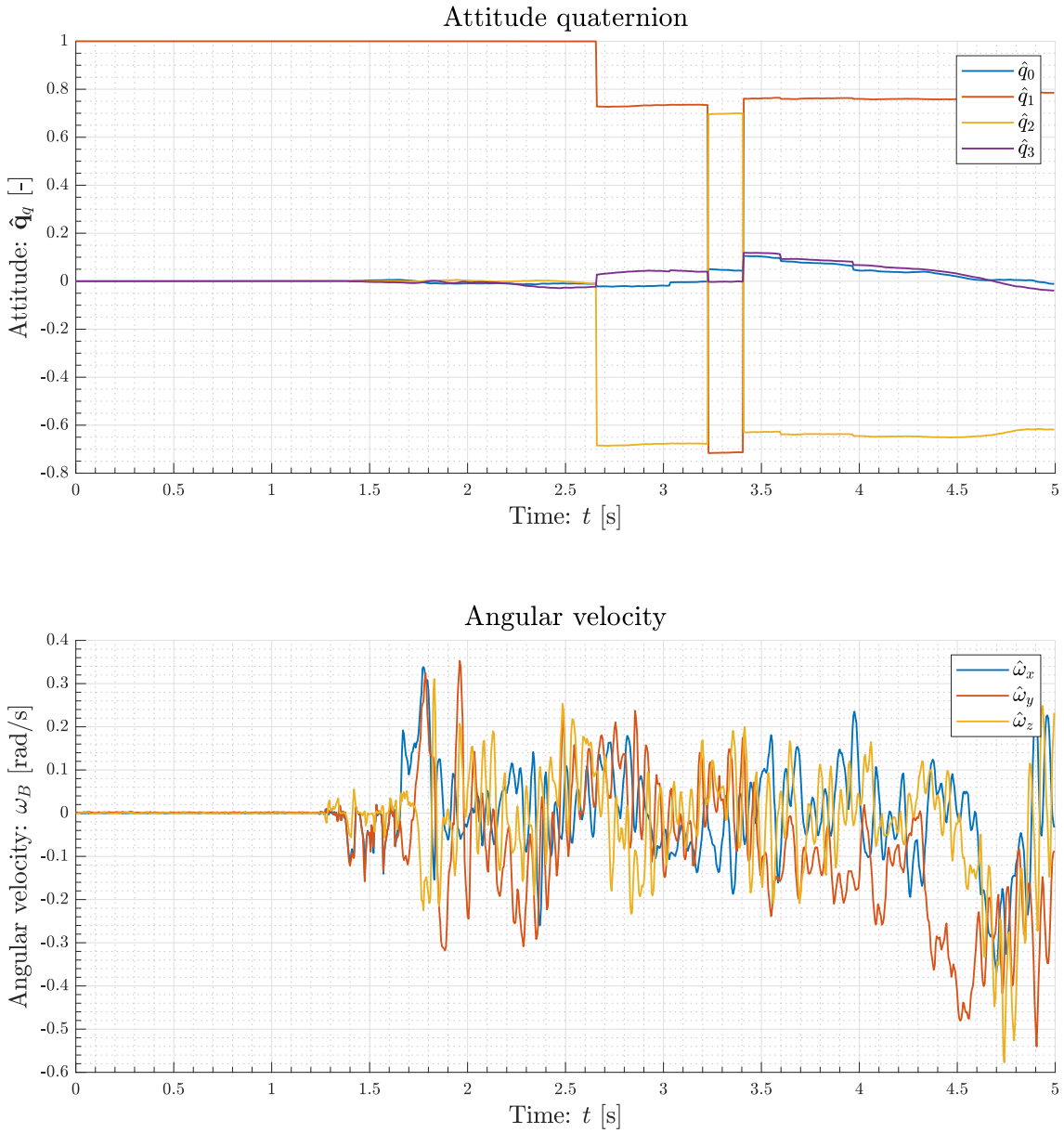



**Figure 6.11:** Estimate of attitude and bias-corrected angular velocity. The large jumps in the estimate occur when the corrupted camera measurements arrive.

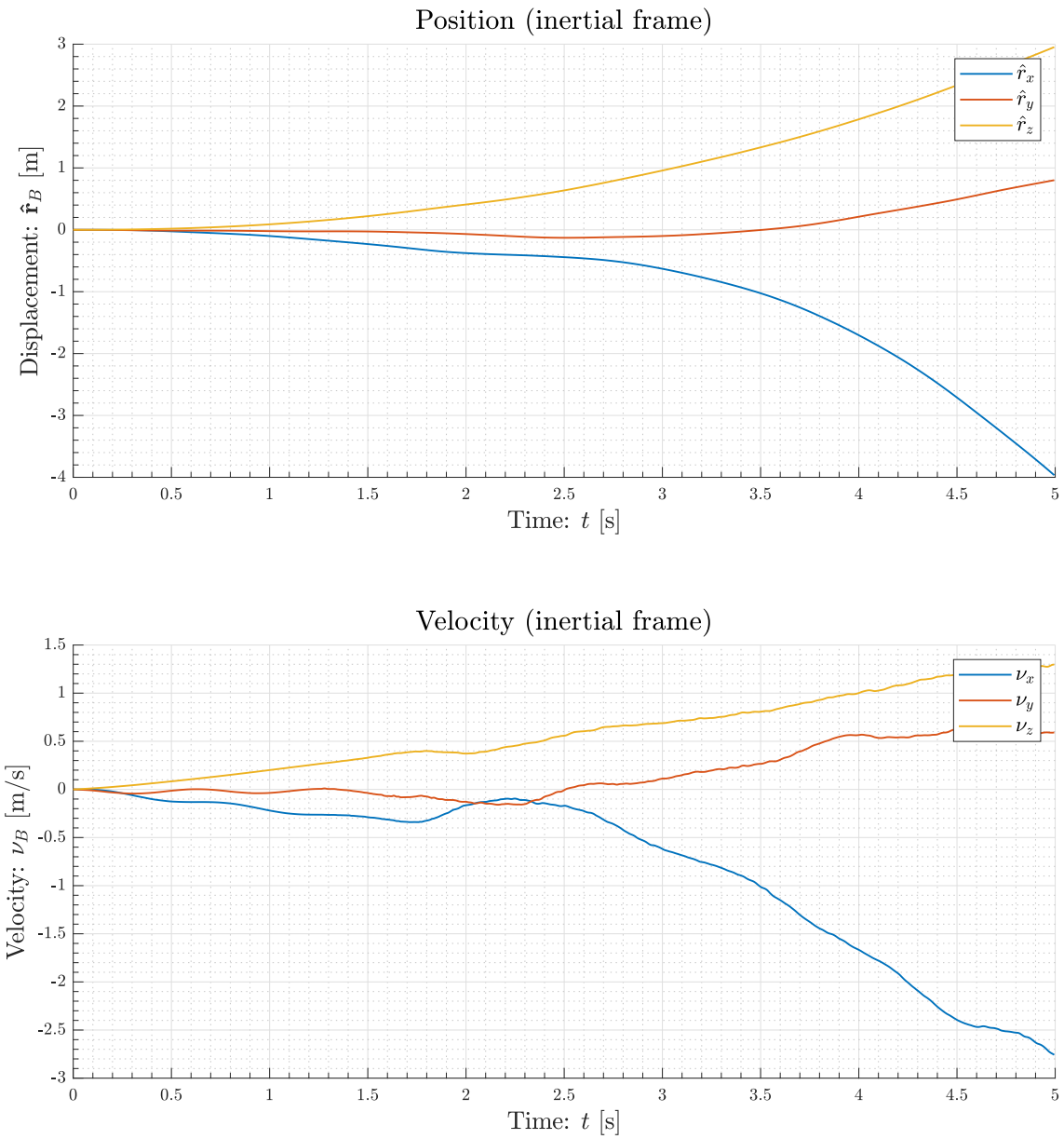



**Figure 6.12:** Estimate of position and velocity. The movements of the sensors was small in this time frame, but due to a lack of useful measurements from the camera, they still provide better estimates.

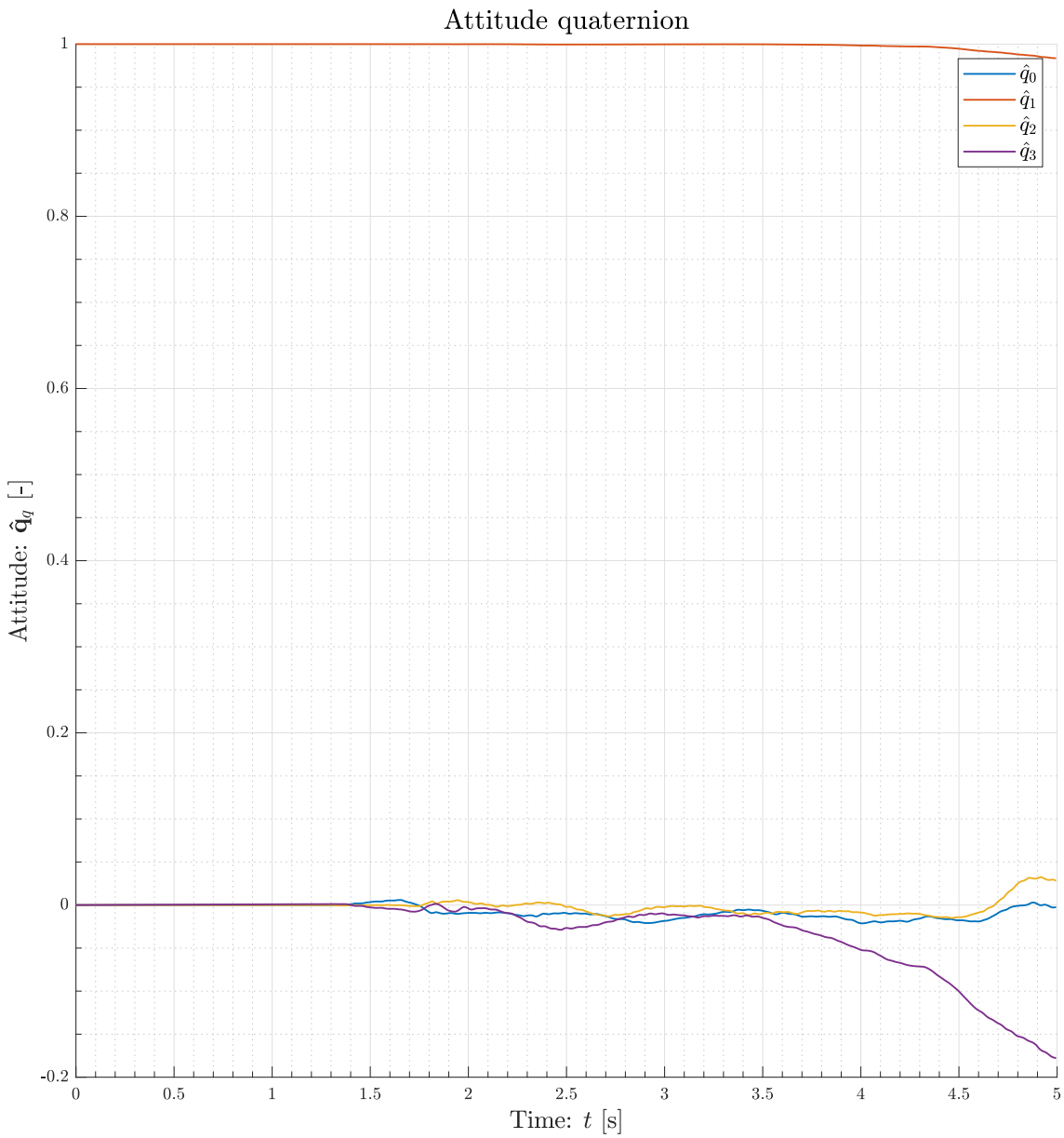



**Figure 6.13:** Estimate of attitude without camera measurements.

# 7 | Discussion & Conclusion

This chapter is dedicated to discussing the observations made during simulation and testing, as well as possible issues, improvements, and future developments.

## 7.1 Discussion

Two different approaches to modelling the system were described in chapter 3, namely the Euler-Lagrange and the Newton-Euler. The latter was not successfully implemented and so the decision to switch to a different approach to modelling was made. The simulation in section 6.1 displays a problem with the model, but it is not the only one. It is however, the one which is most clear in its illustration that the solution to the forward dynamics is broken. Simulating other cases also yield results that are not consistent with reality in some way. Simulating the quadrotor dynamics and the manipulator dynamics in isolation (before attempting to combine them, these simulations were not included in chapter 6 for the sake of brevity) gave the expected outcomes. This points to the error stemming from the solution of the coupled system. It was attempted to be solved symbolically, however due to the complexity of the expressions resulting from the RNE algorithm, the solution was attempted with MATLAB. This same approach was taken for the isolated subsystems and worked well. However, it was evidently not successful in solving the forward dynamics of the UAM. Specialized algorithms for simulating rigid body dynamics for systems with a floating base exist (for example in [30]), but they are complex.

The Euler-Lagrange model was demonstrated in several different cases which were all deemed consistent with the expected outcomes. Model validation with experiments would however strengthen this conclusion significantly. However, several significant physical aspects of the system remain unmodeled. These include:

- **Drag:** Drag on the bodies would dissipate energy from the system and decelerate it. A possible simple model of drag on the UAM would be a model linear in the velocities taking the form:

$$\mathbf{f}_d = \frac{1}{m} \begin{bmatrix} A_x & 0 & 0 \\ 0 & A_y & 0 \\ 0 & 0 & A_z \end{bmatrix} \begin{bmatrix} \dot{r}_x \\ \dot{r}_y \\ \dot{r}_z \end{bmatrix}. \tag{7.1}$$

  Such a model is valid at relatively low speeds, and it is simple enough to not unnecessarily complicate the equations of motion even further [31].

- **Battery voltage:** During flight, the battery voltage will drop, which impacts the estimated parameters, mainly due to a decrease in rotation speed when throttle is held constant and battery voltage drops. This is mainly a concern for the thrust constant. One way to alleviate this is to perform online thrust constant estimation [5]. This may not be necessary depending on the applied controller, but it could also make it possible for the system to automatically adapt its thrust constant when carrying payloads.

- **Rotor dynamics:** In practice, the rotors have dynamics which have not been modelled here. The main components originate from inductance in the motors, inertia of the propeller, and delays from the ESC. A typical approach to modelling rotor dynamics is to simply model it as a first order system:

$$G(s) = \frac{K}{1 + Ts}, \tag{7.2}$$

  where $K$ is the maximum angular velocity of the rotor, and $T$ the time constant. These two parameters can be determined experimentally.

- **Joint friction:** Servos have internal friction which cause them to require extra torque for producing the desired movements. There are many different friction models, but a typical model would be a combination of static and viscous friction [19]. If more accurate models are required, then the Stribeck friction could be included or more advanced models such as Dahl friction could be developed. All of these models require the identification of one or several parameters.

The test of the state estimator showed the camera measurements are unreliable in the current implementation, making it necessary to simply exclude them to get reasonable estimates. However, this causes the integration errors to build up due to the lack of corrections. Including other position or velocity measurements could improve upon this. Possibilities include using GNSS measurements (only for outdoor applications), barometer/altimeter measurements (only for the inertial frame $z$-axis position), optical flow, or indoor positioning systems based on wireless communication systems. However, camera measurements are still expected to be necessary for pick-and-place tasks where the UAM must have very precise position estimates when picking and placing payloads.
It was also noted that camera measurements arrive at a relatively slow rate. Image processing can be computationally expensive but currently the drivers for the specific camera are not directly supported by OpenCV. This causes the transfer of image data from the camera into the OpenCV application programmed for detecting the ArUco markers to be slow, which slows down the entire system. However, this is a software issue which can be fixed. Increasing the amount of available frames should also partially alleviate the issue with motion blur.
Lastly, testing should be done to verify whether it is in fact feasible to use the derived dynamic model in the UKF instead of integrating gyroscope and accelerometer measurements with the IMU-driven model. It would most likely cause trouble to evaluate the model for every single sigma point in the UKF, which is likely to be

too heavy a burden, but estimates may improve.

Due to time constraints caused by the need to redo the modelling with a different approach, a controller for the system was not designed. Due to the complexity of the nonlinear model, designing a controller becomes complicated. Linear controllers would be valid locally but due to the highly nonlinear nature of the model they are unlikely to generalize well. It is possible that repeated linearization around several operating points, possibly combined with a gain scheduling procedure would give good results and has the advantage of not being especially computationally expensive. Nonlinear controllers are also a possible option. Nonlinear control strategies that require manipulating the model equations directly would not be a first choice however.
A nonlinear model predictive controller (NMPC) relying on optimization would be a good candidate as it has the ability to balance the control effort between the rotors and the manipulator joints in an optimal sense. This would allow the UAM to efficiently use its manipulator joint to control not only the manipulator joint states, but also the quadrotor states. The prediction of future states within the finite horizon is also an advantage, as the NMPC practically computes a trajectory. A downside is that it requires evaluating the model functions many times at every time step due to the finite horizon. For this reason the computational load becomes substantial, especially when combined with an optimization algorithm, and the control algorithm may become so slow that it is not feasible to use in real time. There is also the concern that since the model is highly nonlinear, it can be assumed to be non-convex which means that optimization solvers may not be able to find the global minima and may get stuck. Controllers that are slow to evaluate could be complemented by simpler and faster trajectory tracking controllers which could make the UAM follow the trajectory computed by the NMPC. This is deemed to be a good approach that could be attempted in the future.
It is also possible to take a model-free approach to the control and train a machine learning model with reinforcement learning to find an optimal control law. The model could be used for simulating the system which could be used as a starting point for doing most of the (initial) training. Further training would most likely be necessary on the physical system, to ensure that the learning agent would also be able to learn the unmodelled dynamics.

## 7.2 Conclusion

A prototype UAM has been designed and assembled, including sensing and processing devices. Two separate dynamic models have been investigated and simulated to verify the results. Only the Euler-Lagrange model showed acceptable results in simulation, while the Newton-Euler model, based on the Recursive Newton-Euler algorithm failed due to issues with solving the coupled dynamic equations. This is likely possible to solve with specialized algorithms for solving this type of system. The Euler-Lagrange model provides closed form expressions for the dynamics, but the equations of motion are very complex, and are further complicated by the need to handle the quaternion constraint. Despite this, several unmodelled dynamics

such as drag, joint friction, and rotor dynamics could be included to make it a more accurate representation of reality. A state estimator for the UAM was designed and implemented. It was made as an adaptation of the Unscented Quaternion Estimator, to also estimate states other than the attitude and gyro bias. Due to corrupted camera measurements the estimates were better when simply integrating the gyro and accelerometer measurements.
Due to time constraints - mainly caused by the necessity to redo the dynamic model in the Lagrangian formulation - a controller was not designed, but as possible future control designs NMPC or reinforcement learning-based approaches are suggested due to their ability to handle the highly nonlinear nature of the system without relying on linearization or manipulation with the equations of motion.
Future work includes control design, and correcting the camera measurement system to be able to use the measurements for state estimation.

# Bibliography


[1] Hossein Bonyan Khamseh, Farrokh Janabi-Sharifi, and Abdelkader Abdessameud. "Aerial manipulation—A literature survey". eng. In: *Robotics and autonomous systems* 107 (2018), pp. 221–235. ISSN: 0921-8890.

[2] Ling Li, Tianlin Zhang, Hang Zhong, et al. "Autonomous Removing Foreign Objects for Power Transmission Line by Using a Vision-Guided Unmanned Aerial Manipulator". eng. In: *Journal of intelligent & robotic systems* 103.2 (2021). ISSN: 0921-0296.

[3] Gowtham Garimella and Marin Kobilarov. "Towards model-predictive control for aerial pick-and-place". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 4692–4697. DOI: 10.1109/ICRA.2015.7139850.

[4] Airflight ApS. *Building Tomorrow's High Precision Flying Cranes*. URL: https://www.airflight.io/.

[5] Gowtham Garimella, Matthew Sheckells, Soowon Kim, et al. *A Framework for Reliable Aerial Manipulation*. 2018.

[6] Justin Thomas, Giuseppe Loianno, Koushil Sreenath, et al. "Toward image based visual servoing for aerial grasping and perching". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 2113–2118. DOI: 10.1109/ICRA.2014.6907149.

[7] Yanjie Chen, Yangning Wu, Zhenguo Zhang, et al. "Image-based Visual Servoing of Unmanned Aerial Manipulators for Tracking and Grasping a Moving Target". In: *IEEE Transactions on Industrial Informatics* (2022), pp. 1–11. DOI: 10.1109/TII.2022.3222688.

[8] Daniel Mellinger, Quentin Lindsey, Michael Shomin, et al. "Design, modeling, estimation and control for aerial grasping and manipulation". eng. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2668–2673. ISBN: 1612844545.

[9] Matko Orsag. *Aerial Manipulation*. eng. 1st ed. Advances in Industrial Control. Cham: Springer International Publishing, 2018. ISBN: 3-319-61022-8.

[10] Matko Orsag, Christopher Michael Korpela, Stjepan Bogdan, et al. "Hybrid Adaptive Control for Aerial Manipulation". eng. In: *Journal of intelligent & robotic systems* 73.1-4 (2014), pp. 693–707. ISSN: 0921-0296.

[11] Bin Yang, Yuqing He, Jianda Han, et al. "Rotor-Flying Manipulator: Modeling, Analysis, and Control". eng. In: *Mathematical problems in engineering* 2014 (2014), pp. 1–13. ISSN: 1024-123X.

[12] Hossein Bonyan Khamseh and Farrokh Janabi-Sharifi. "UKF-Based LQR Control of a Manipulating Unmanned Aerial Vehicle". In: *Unmanned Syst.* 5 (2017), pp. 131–139.

[13] Le Ma, Yiming Yan, Zhiwei Li, et al. "A novel aerial manipulator system compensation control based on ADRC and backstepping". eng. In: *Scientific reports* 11.1 (2021), pp. 22324–22324. ISSN: 2045-2322.

[14] Georgios Darivianakis, Kostas Alexis, Michael Burri, et al. "Hybrid predictive control for aerial robotic physical interaction towards inspection operations". eng. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014. ISBN: 1479936855.

[15] Michail Kalaitzakis, Brennan Cain, Sabrina Carroll, et al. "Fiducial Markers for Pose Estimation: Overview, Applications and Experimental Comparison of the ARTag, AprilTag, ArUco and STag Markers". In: *Journal of Intelligent Robotic Systems* 101 (Apr. 2021). DOI: 10.1007/s10846-020-01307-9.

[16] Paul Kremer, Jose Luis Sanchez-Lopez, and Holger Voos. "A Hybrid Modelling Approach for Aerial Manipulators". In: *Journal of Intelligent &amp Robotic Systems* 105.4 (2022). DOI: 10.1007/s10846-022-01640-1. URL: https://doi.org/10.1007%2Fs10846-022-01640-1.

[17] Hannes Sommer, Igor Gilitschenski, Michael Bloesch, et al. *Why and How to Avoid the Flipped Quaternion Multiplication*. 2018. arXiv: 1801.07478 [cs.RO].

[18] F. Landis Markley and John L Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. eng. Vol. 33. Space Technology Library. New York, NY: Springer, 2014. ISBN: 9781493908011.

[19] J.J. Craig. *Introduction to Robotics: Pearson New International Edition PDF eBook: Mechanics and Control*. Pearson Education, 2013. ISBN: 9781292052526. URL: https://books.google.dk/books?id=ZTqpBwAAQBAJ.

[20] Howie Choset, Kevin M. Lynch, Seth Hutchinson, et al. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. English. MIT Press, May 2005. ISBN: 0262033275.

[21] Basile Graf. *Quaternions and dynamics*. 2008. arXiv: 0811.2889 [math.DS].

[22] OpenCV. *ArUco marker detection*. URL: https://docs.opencv.org/4.5.1/d9/d6d/tutorial_table_of_content_aruco.html.

[23] Robotis Co. Ltd. *XH430-W210 Documentation*. 2023. URL: https://emanual.robotis.com/docs/en/dxl/x/xh430-w210/.

[24] STMicroelectronics N.V. *LSM6DSOX*. 2023. URL: https://www.st.com/en/mems-and-sensors/lsm6dsox.html.

[25] John Crassidis. "Unscented Filtering for Spacecraft Attitude Estimation". In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 26 (July 2003). DOI: 10.2514/2.5102.

[26] Rudolph Van der Merwe and Eric Wan. "The Square-Root Unscented Kalman Filter for State and Parameter-Estimation". In: vol. 6. Feb. 2001, 3461 –3464 vol.6. ISBN: 0-7803-7041-4. DOI: 10.1109/ICASSP.2001.940586.

[27] Mohinder S Grewal and Angus P Andrews. *Kalman filtering: theory and practice using MATLAB*. eng. 3rd ed. Wiley, 2008. ISBN: 9780470377802.

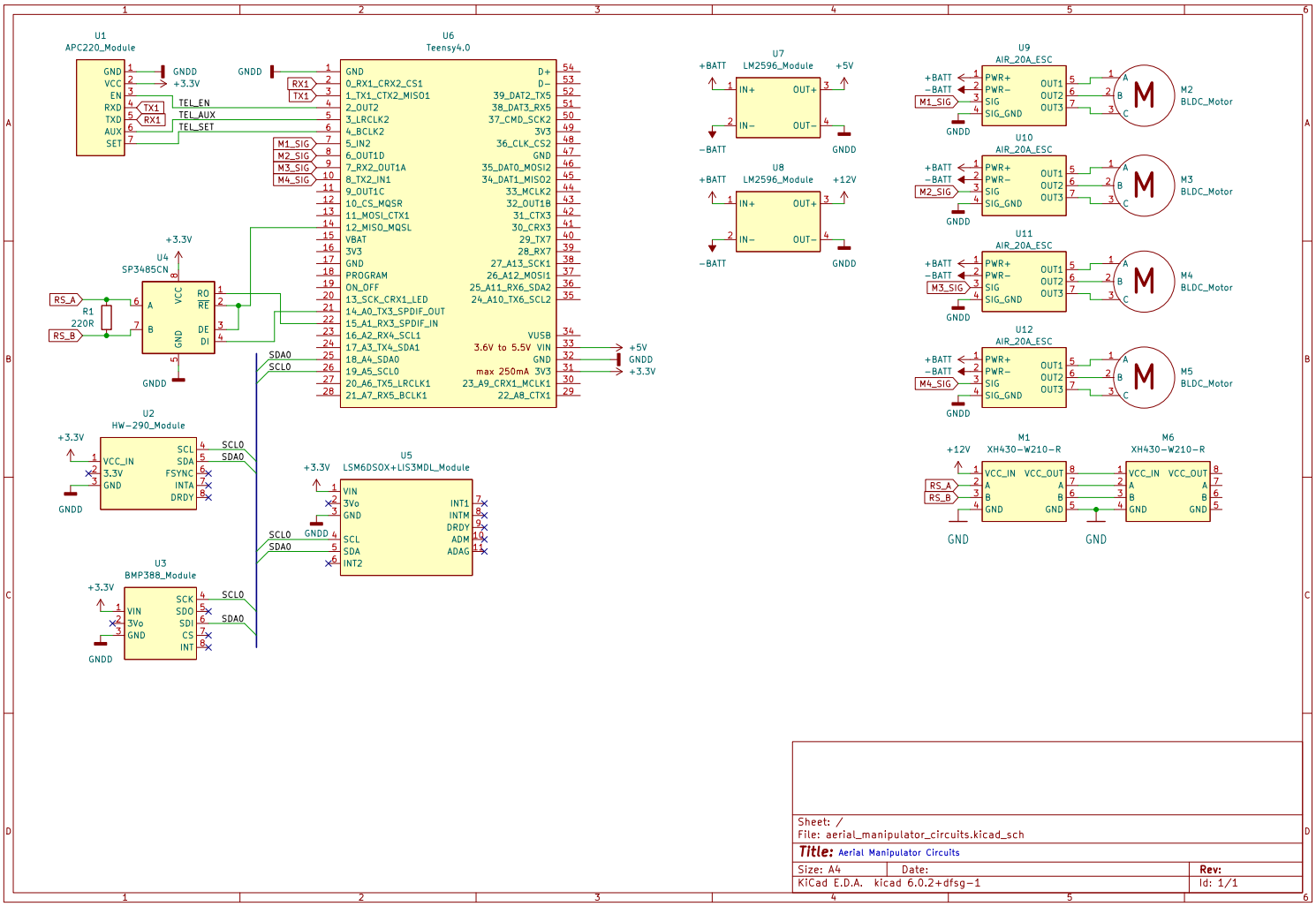
[28] Landis Markley. "Attitude Error Representations for Kalman Filtering". In: *Journal of Guidance Control and Dynamics - J GUID CONTROL DYNAM* 26 (Mar. 2003), pp. 311–317. DOI: 10.2514/2.5048.

[29] Joan Solà. *Quaternion kinematics for the error-state Kalman filter*. 2017. arXiv: 1711.02508 [cs.RO].

[30] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Vol. vol. 49. Jan. 2007. ISBN: 978-1-4757-6437-6. DOI: 10.1007/978-0-387-74315-8.

[31] Teppo Luukkonen. *Modelling and control of quadcopter*. Aalto University, Aug. 2011.

# A | Electrical Connections




| | | |
|---|---|---|
| Sheet: / | | |
| File: aerial_manipulator_circuits.kicad_sch | | |
| **Title:** Aerial Manipulator Circuits | | |
| Size: A4 | Date: | **Rev:** |
| KiCad E.D.A. kicad 6.0.2+dfsg-1 | | Id: 1/1 |