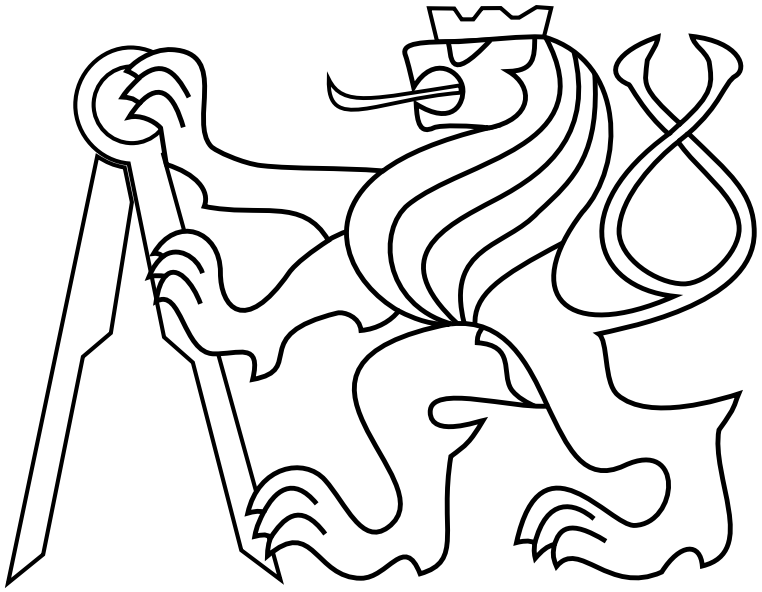
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

BACHELOR'S THESIS



Jiří Horyna

Robotic manipulation onboard an unmanned aerial vehicle

Department of Cybernetics

Thesis supervisor: **Ing. Tomáš Báča**

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Horyna Jiří** Personal ID number: **457177**

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

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Robotics**

II. Bachelor's thesis details

Bachelor's thesis title in English:

Robotic Manipulation Onboard an Unmanned Aerial Vehicle

Bachelor's thesis title in Czech:

Robotická manipulace z paluby bezpilotní helikoptéry

Guidelines:

The work aims to incorporate a robotic manipulator into the Unmanned Aerial Vehicle (UAV) platform used in the MRS lab at FEE and is structured as follows:
1. The student will design a prototype of a simple 2-DOF manipulator, which will serve as a proof of concept for the methods tackled in work. The hardware will be integrated within the current software architecture of the UAV in the form of a program in Robot Operating System (ROS) platform. The program will allow controlling the manipulator (DKT, IKT) from the UAV's onboard computer.
2. The manipulator will be modeled in the realistic Gazebo simulator as well as assembled as a hardware prototype.
3. The dynamical model of the UAV will be extended with the model of the manipulator, and influence of the manipulator on the performance of the onboard position controller will be identified.
4. The student will design a method for mitigation of the predicted disturbance to minimize the manipulation error during the flight.The proposed method will be thoroughly tested in the Gazebo simulator using the lab's ROS software platform.
5. Limits in the accuracy of the manipulation should be found to direct the future development of the topic.

Bibliography / sources:

[1] Kim, Suseong, Seungwon Choi, and H. Jin Kim. "Aerial manipulation using a quadrotor with a two dof robotic arm.", International Conference on Intelligent Robots and Systems (IROS), IEEE, 2013.
[2] T. Baca, "Embedded Model Predictive Control of Unmanned Aerial Vehicle", Master's thesis, CVUT FEL, 2015.
[3] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

Name and workplace of bachelor's thesis supervisor:

Ing. Tomáš Báča, Multi-robot Systems, FEE

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

Date of bachelor's thesis assignment: **12.01.2018** Deadline for bachelor thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Tomáš Báča	doc. Ing. Tomáš Svoboda, Ph.D.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

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

. _____
Date of assignment receipt

_____
Student's signature

Author statement for undergraduate thesis

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

Prague, date..... .....

signature

Acknowledgements

Firstly I would like to thank my supervisor Tomáš Báča for his great support during this thesis. Furthermore I thank other members of Multi-robot Systems group for their advices. I would also like thank to my family for providing me the opportunity to study.

Abstract

This thesis deals with design, construction and control of a manipulator onboard on Unmanned Aerial Vehicle (UAV). Aerial manipulation introduces disturbances in control of the UAV, which if not addressed, decreases precision of the flight and therefore of the manipulation. This work aims to design and test a method to mitigate control disturbances caused by the manipulator in real time. Simulated experiments found limits in control of the system. Possibilities of use were verified on a hardware prototype of the robotic arm.

Keywords: aerial manipulation, UAV, system control, ROS, Gazebo

Abstrakt

Tato práce se zabývá návrhem, konstrukcí a řízením manipulátoru z paluby bezpilotní helikoptéry. Manipulace za letu způsobuje poruchy v řízení bezpilotní helikoptéry, které pokud nejsou žádoucí, snižují přesnost letu a tím i přesnost manipulace. Tato práce je zaměřena na návrh a otestování techniky, která by zmírnila poruchy v řízení způsobené manipulátorem v reálném čase. Simulované experimenty ukázaly limity v řízení tohoto systému. Na prototypu robotického ramene byly ověřeny možnosti jeho použití.

Klíčová slova: manipulace za letu, UAV, řízení systému, ROS, Gazebo

Contents

List of Figures

1 Introduction

The Unmanned Aerial Vehicle (UAV) is an aircraft without pilot on board. Thus, it can be controlled remotely by an operator or its flight is autonomous. In this thesis, the term UAV refers to a multicopter. Multicopter is a symmetrical vehicle with brushless motors, which directly drive the same number of propellers. By control of individual motors (thrust of propellers) we can determine tilt of the vehicle. Nowadays, multicopters are widely used. The reason may be their decreasing price and multipurpose usage. They can find application in military, research, film making, hobby and even in sport (drone racing).

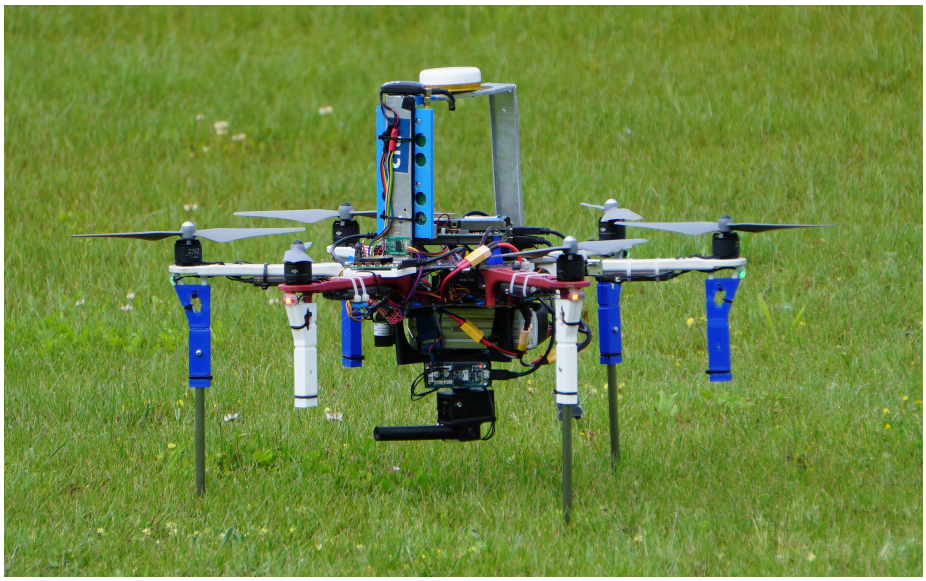


Figure 1: A hexacopter from MRS laboratory equipped with assembled manipulator.

When manipulate with some object by an aerial vehicle, we conduct aerial manipulation. Recently, this topic became attractive in research. There are several methods of aerial manipulation which differ from each other by type of a grasp or type of a manipulator. The first approach is to attach a gripper directly under the UAV. In this case we have to consider weight of the payload and shift of the center of mass from its original location. However, the gripper with a payload has fixed position, which makes control easier. The second approach is to substitute fixed gripper with towed cables or with solid link supplemented by passive spherical joint. This is mainly used in cooperative aerial manipulation. The last main approach is to equip an aerial vehicle with manipulator with active joints. The UAV equipped with such manipulator has several effects on the whole system. Specifically it is the floating center of mass, a variable inertia tensor and emergence of forces generated by manipulator's joints motions. These effects are not desirable and cause position errors. We use two methods for control position errors. The first one is that we see the problem as two system. In this case, we consider disturbances caused by manipulator as external disturbances, which we are trying to mitigate. The second approach is to consider

the entire system (the UAV + the manipulator). This way is much more difficult, because precise and complex dynamic model have to be known.

Aerial manipulators have wide spectrum of use in many areas of use. They increases a work space of a fixed robotic arm, thus the manipulator can be used even on hard to reach places. Here, the thesis do not aim on manipulation with object as such. Instead of that, influences of the manipulation, their avoidance or eventually their use will be examined.

1.1 Problem statement

The task of this thesis is to design and construct a manipulator prototype that will be added to the UAV. The construction will be performed for both, the realistic Gazebo simulator and the real UAV. Motions of the manipulator, especially if it is loaded, cause errors in position of the UAV. Firstly, sources of these errors will be described. It is followed by identification of influence of these sources on the UAV's position. The appropriate method of control will be designed. The outcome of this thesis should be testing of designed controller and find out possibility of use of the hardware prototype.

1.2 State of the art

Aerial manipulation was already tested by MRS[1] group before [7]. They designed lightweight magnetic gripper, which allows to carry flat ferrous objects. Their gripper was equipped with two Hall sensors to identify a successful grasping. Test results shows, that they are able to localize, grasp and transport object with high success rate (around 90%). Their control method is so robust, that position errors during flight with payload are negliable.

A multilayer control of an UAV equiped with manipulator is desdigned in [18]. First layer of the proposed method uses a UAV's battery on a linear slider as counterweight. The center of mass of the system is kept near to the UAV's geometric center. This control method has limits such as speed of the battery movement. The second control layer, where thrust of individual propellers is controlled in software, takes care about control above the limits of the first layer. In the last layer, they estimate dynamic effects of the manipulator on the UAV. These estimations are then brought to the controller of this estimated effects. They have done several test flights with the manipulator, where individual control layers were gradually added. Position error was reduced to about 21% its original value, while all control layers were used.

A different approach in this area of aerial manipulation is about the design of the controller, which is based on dynamic model of the whole system (an UAV equipped with

[1] Multi-robot Systems, http://mrs.felk.cvut.cz/

a robotic arm). A research of this type is conducted at the University of Seville at the Robotics, Vision and Control Group [3, 4]. Creation of the combined model is very complex and it is described by Newton-Euler dynamic equations in those papers. A manipulator arm with 7 degrees of freedom with maximum payload 1.5 kg is used in [3]. In [4], lighter type of manipulator was developed by University of Seville and CATEC[2]. In both cases, a nonlinear controller (backstepping controller with an integral term) is used. They tested designed method of control in both, simulations and real outdoor experiments. The nonlinear controller has achieved much better results (UAV's Euler angles were more stable), then PID controller used before.

Advanced branch of aerial manipulation is cooperative manipulation, where several UAVs grasp a single object. In [9] a human operator drives group of quadrocopters in simulation. Manipulators are represented by passive arm with spherical joint for this operation. Grasping is solved by towed cables in [8] and [20]. Advantage of cooperative aerial manipulation is the possibility of lifting bigger payloads. In case of grasping by cables, they are able to control orientation of the grasped object, which is not simple task in case of one aerial vehicle. Another papers with methods similar to the presented ones are [22], [5] and [12].

1.3 Contributions

We present a 2-DOF (Degree of freedom) manipulator in a form of a Gazebo model and in a form of hardware prototype. The hardware prototype was integrated within the current software architecture of the UAV, thus it can be controlled onboard of the UAV during flight. Designed aerial manipulator is able to grasp object with higher accuracy thanks the fact, that it was designed as manipulator with active joints. Manipulation with heavy payloads (around 0.5 kg) was tested in the Gazebo simulator. Errors caused by the manipulator during these test flights were identified and controlled by method of Euler angles adjustment. We get satisfactory result (improvement of error position approximately 40%), when we take into account, that we implemented only one control layer and we consider our aerial manipulator as two separated system (system of the UAV and system of the manipulator).

1.4 Outline

At first, a main software platforms is introduced in this thesis. Further, a manipulator's structure is described in section 3 with an emphasis on forward and inverse kinematics. This is followed by section 4, where sources of UAV's disturbances caused by manipulator are computed. Modeling of the manipulator for Gazebo simulator is described in section 5

[2]Center for Advanced Aerospace Technologies

as well as description of hardware prototype. In section 6 the dynamic model of the UAV is presented. It is followed by modification of this dynamic model (section 6.2), so as to describe system of the UAV equipped with the manipulator. Furthermore, identification of influences of manipulator's motions on the UAV is estimated (section 7). An appropriate method of control of the UAV to mitigate identified disturbances is designed and tested (section 8).

2 Preliminaries

Current software architecture of the UAV is implemented in Robot Operating System (ROS) platform. Programs, which were created during this thesis, follow up on this software architecture. Furthermore, the manipulator will be modeled in the realistic Gazebo simulator, where the proposed method of control of the UAV will be tested. ROS and the Gazebo simulator are presented in this section.

2.1 Robot Operating System

ROS [11, 14] is an open-source, middleware system running on Unix-based platforms. Thus, it provides services above the host operating system[3]. These services help software developers to create robot systems. The basis of ROS is packages[4], where all software, including our software, is located. Executable and supporting files are implemented in those packages. Another term is node[5], a running instance of program in ROS. Nodes communicate with each other through messages[6] or service calls[7] (figure 2). A node, which provides an information publishes message to a topic[8]. On the other hand, receiving nodes subscribe those messages. Service calls are bi-directional and supports only one-to-one communication unlike messages. Numerous versions of ROS distributions[9] were released. The newest one is called Lunar Loggerhead, however the Kinetic Kame distribution is used here.

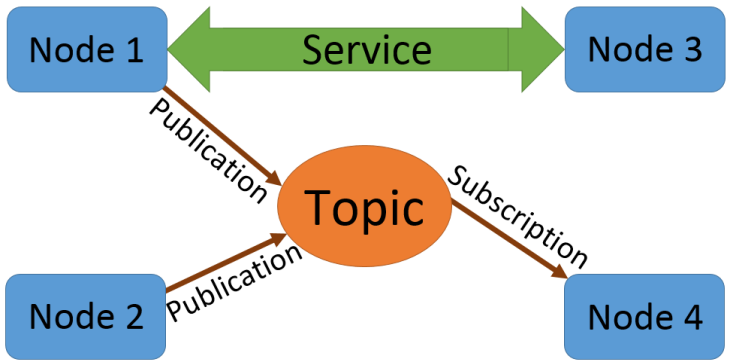


Figure 2: Diagram of the communication between nodes in ROS.

[3]Ubuntu 16.04 in our case.
[4]http://wiki.ros.org/Packages
[5]http://wiki.ros.org/Nodes
[6]http://wiki.ros.org/Messages
[7]http://wiki.ros.org/Services
[8]http://wiki.ros.org/Topics
[9]ROS distribution is a version of set of ROS packages.

2.2 Gazebo simulator

Gazebo [2] is a 3D simulator, whose greatest benefit is testing of new robotics algorithms without need of real devices. It allows us test algorithms anytime and without risk of damage of the tested device. New robots can be designed and tested before prototyping and production. Several virtual environments are prepared for users, however custom worlds and scenarios can be created. Gazebo is open-source and has many contributors, who are still evolving the simulator. Gazebo simulator was integrated to ROS, creating a powerful tool for development and testing of robots. User interface of the Gazebo simulator is shown in figure 3.

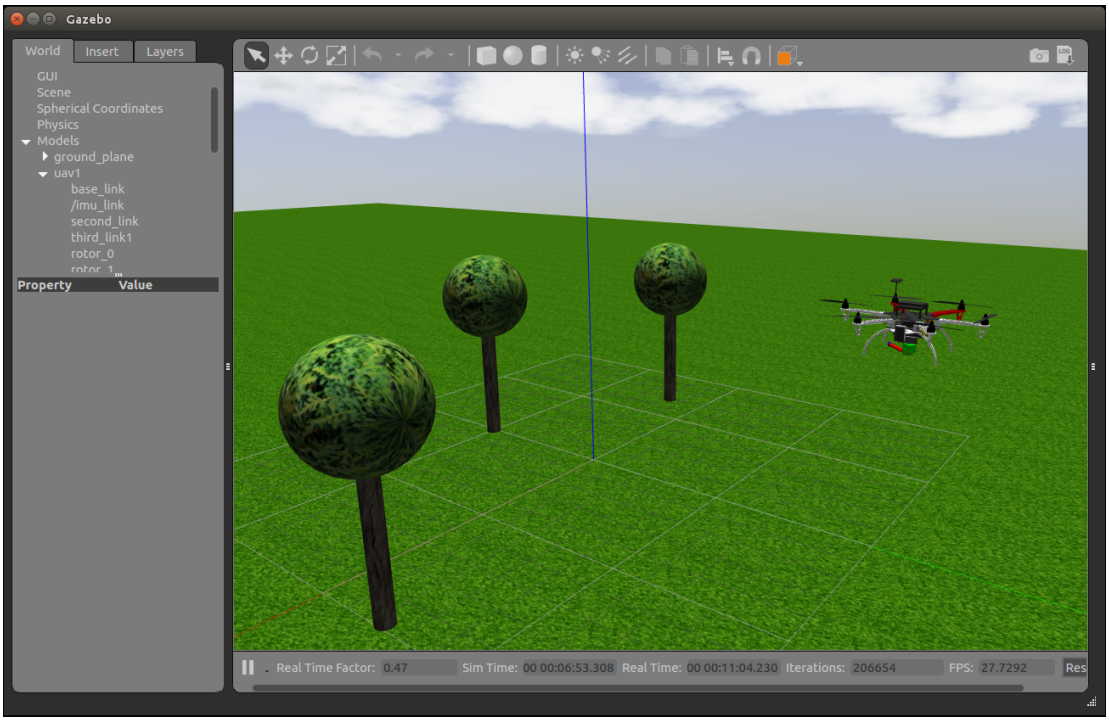


Figure 3: User interface of the Gazebo simulator showing a flying Unmanned Aerial Vehicle.

3 Description of the manipulator

Due to the fact that the main subject of this task is control of a UAV, the manipulator was designed as a device with ordinary specifics. The design of the arm is shown in figure 4 and specifics of the manipulator are summarized in following points:

- Open kinematic chain - kinematic chain of the designed manipulator can be represented by acyclic graph.
- Assembled with two revolute joints, where axis of rotation of the first joint is identical to z-axis of manipulator's coordinate system. Axis of rotation of the second joint is perpendicular to the axis of rotation of the first joint.
- Two degrees of freedom (2-DOF) - DOF is represented by a number of independent parameters of manipulator needed to specify the position of the mechanism in given coordinate system. These parameters is the pair of joint coordinates θ_1 and θ_2.
- Due to the manipulator's construction described above, the operational space (subspace of the ambient space E^3 occupied by any of the robot part during any of possible manipulator motions) is a hemisphere under the frame of UAV. The work envelope (subspace of operational space where the manipulator can reach by the end-effector) is the surface of that hemisphere.

3.1 Coordinate system of the manipulator

Coordinate system of the manipulator (M) is translated relative to the coordinate system of the UAV (B) by an offset l_1 in the direction of the $z^{(B)}$ axis. The reason of setting this coordinate system is an easier description of the manipulator while all the attributes are retained. Let $s^{(B)}$ and $s^{(M)}$ be set as homogeneous coordinates in coordinate systems B and M. Then transfer relationship between these systems is

$$\mathbf{s}^{(M)} = \mathbf{T}_z(l_1)\mathbf{s}^{(B)}, \tag{1}$$

where $\mathbf{T}_z(l_1)$ is translation matrix with translation along z-axis of length l_1.

3.2 Forward kinematics

Forward kinematics is a mapping from joint coordinate space to space of end-effector positions. Thus, we know the position of individual joints by direct measuring of the joint coordinates and we calculate coordinates of the end-effector in Cartesian coordinate system

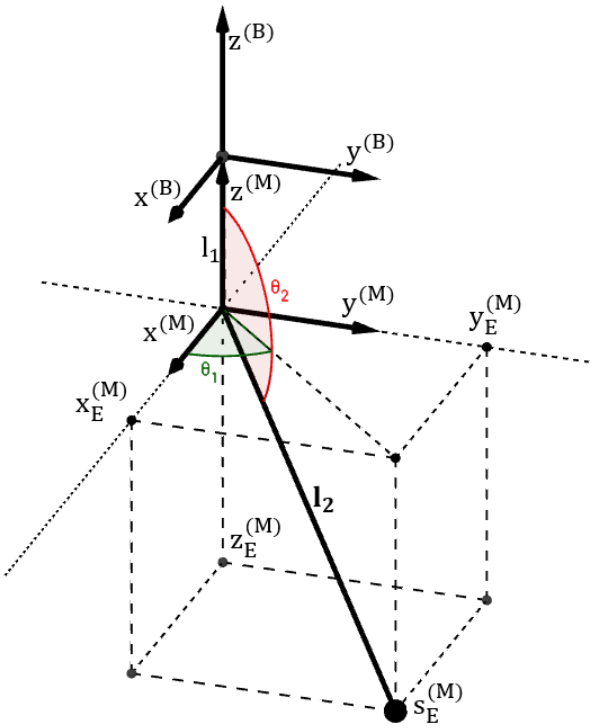


Figure 4: Design of the manipulator with assigned parameters which are used for further description.

of the manipulator. Let vector $\mathbf{q} = \binom{\theta_1}{\theta_2}$ be set as joint coordinates. Then mapping of forward kinematics is

$$\mathbf{q} \to \mathbf{s}_E^{(M)}, \tag{2}$$

where s_E^M are coordinates of the end-effector in manipulator's coordinate system. For this task it is necessary to know forward kinematics mainly for using simulator, where the end-effector needs to be tracked during its motions. Forces, acting on the UAV are then calculated based on the position of the end-effector. For such manipulator, which was designed, the forward kinematics is similar to conversion from spherical coordinates to Cartesian coordinates. Thus, the equations follow the form

$$\begin{aligned} x_E^{(M)} &= l_2 \cos\theta_1 \sin\theta_2, \\ y_E^{(M)} &= l_2 \sin\theta_1 \sin\theta_2, \\ z_E^{(M)} &= l_2 \cos\theta_2, \end{aligned} \tag{3}$$

where $\theta_1 \in [0, 2\pi)$ and $\theta_2 \in [\frac{\pi}{2}, \frac{3\pi}{2})$.

3.3 Inverse kinematics

Inverse kinematics is a mapping from space of end-effector positions to joint coordinate space. This mapping is inverse to the forward kinematics, so we can write

$$\mathbf{s}_E^{(M)} \rightarrow \mathbf{q}. \tag{4}$$

Due to this function returns angels of rotations of individual joints, inverse kinematics will be used for planning of end-effector trajectories. Specific joint coordinates can be calculated as

$$\begin{aligned} \theta_{11} &= atan2(y_E^{(M)}, x_E^{(M)}), \\ \theta_{21} &= \arccos \frac{z_E^{(M)}}{l_2}, \end{aligned} \tag{5}$$

where $l_2 = \sqrt{{x_E^{(M)}}^2 + {y_E^{(M)}}^2 + {z_E^{(M)}}^2}$. Inverse kinematics task for such manipulator has two solutions. The second one is

$$\begin{aligned} \theta_{12} &= \theta_{11} + \frac{\pi}{2}, \\ \theta_{22} &= -\theta_{21}. \end{aligned} \tag{6}$$

4 Action of forces on the UAV caused by the manipulator

Several forces acting on UAV are created by attaching the manipulator or by motions of the manipulator's links. These forces are sources of disturbances of flying UAV. We need to know magnitudes, directions and points of application of these forces to be able to simulate behavior of UAV in the simulator. Forces, which has not negligible impact on movement of the UAV include the force of gravity, centrifugal force and Euler force. For further description of presented forces we have to introduce mass of individual links of robot. Let them be introduced as follows:

- m_1 is the mass of the base of manipulator and its first link described with length l_1,
- m_2 is the mass of the second link l_2,
- m_3 is the mass of weight located on the position of the end-effector.

Further we need to express positional vectors as well. Let $\mathbf{l}_2$ be vector which describes position of end-effector in the coordinate system M and $\mathbf{r}_2$ be vector describing position of the end-effector in xy-plane. Mathematically it will be

$$\begin{aligned} \mathbf{r}_2 &= \begin{pmatrix} x_E & y_E & 0 \end{pmatrix}^{(M)}, \\ \mathbf{l}_2 &= \begin{pmatrix} x_E & y_E & z_E \end{pmatrix}^{(M)}. \end{aligned} \tag{7}$$

I decided to use the same positional vectors $\mathbf{r}_2$ and $\mathbf{l}_2$ for each link due to the fact that the direction of forces for each link is equal and magnitudes of forces can be adjusted by multiplying by a constant. Points of application of forces, which will be analyzed in following sections, are located in center of gravity of each link.

4.1 Force of gravity

Force of gravity (weight) acts at all times on all object close to earth surface thus including links of the manipulator. Due to this, the first impact on the UAV is decrease of the altitude. Further by adding an extra mass to UAV a change of position of the center of gravity occurs. These changes are one of the sources which has impact on the tilt of drone. Direction of the weight is the same as direction of the acceleration caused by gravity – downward toward the center of Earth

$$\mathbf{F}_{\mathbf{G}i} = m_i \mathbf{g}, \tag{8}$$

where $i \in \{1, 2, 3\}$.

4.2 Centrifugal force

Centrifugal force is an inertial force directed outwards from the rotating object and center of rotation in axis of radius of the rotation. A magnitude of this force depends on angular velocity $\dot{\theta}$ as well. With the manipulator onboard, the centrifugal force is acting just while the manipulator is in motion in its coordinate system. We describe this force for movements in each joint separately. The summary of them is in Table 1.

Table 1: Centrifugal forces acting on the UAV

	Motion of the first joint	Motion of the second joint
First link	$\mathbf{F}_{C21} \approx 0$	$\mathbf{F}_{C12} = 0$
Second link	$\mathbf{F}_{C11} = \frac{1}{2} m_2 \dot{\theta}_1^2 \mathbf{r}_2$	$\mathbf{F}_{C22} = \frac{1}{2} m_2 \dot{\theta}_2^2 \mathbf{l}_2$
Third link (weight)	$\mathbf{F}_{C31} = \frac{1}{2} m_3 \dot{\theta}_1^2 \mathbf{r}_2$	$\mathbf{F}_{C32} = \frac{1}{2} m_3 \dot{\theta}_2^2 \mathbf{l}_2$

We consider that the first joint rotates with the first link around the first link's center, thus $\mathbf{F}_{C21} \approx 0$.

4.3 Euler force

In presence of angular acceleration $\ddot{\theta}$, the Euler force exists. It means that this force acts on the UAV just for the moment while drives of joints accelerate or decelerate. Direction of Euler force is given by vector product of angular acceleration and positional vector (to clarify idea, the direction can be described as opposite to direction of tangential acceleration). Euler forces for each link are presented in following table. We consider $\mathbf{F}_{E11} \approx 0$ due to the same reason as in the case of the centrifugal force

4.4 Resulting force and torque

After analysis of individual forces, we need to substitute these forces by their resulting force and torque (moment), for the simulation described in section 4.1. Due to the fact that forces acting on the UAV have rotating effect, we have to relate them to the pivot point. The object (the UAV) is not fixed, thus the pivot point is located in the center of mass

Table 2: Euler forces acting on the UAV

	Motion of the first joint	Motion of the second joint
First link	$\mathbf{F}_{E11} \approx 0$	$\mathbf{F}_{E12} = 0$
Second link	$\mathbf{F}_{E21} = -\frac{1}{2} m_2 \ddot{\theta}_1 \times \mathbf{r}_2$	$\mathbf{F}_{E22} = -\frac{1}{2} m_2 \ddot{\theta}_2 \times \mathbf{l}_2$
Third link (weight)	$\mathbf{F}_{E31} = -\frac{1}{2} m_3 \ddot{\theta}_1 \times \mathbf{r}_2$	$\mathbf{F}_{E32} = -\frac{1}{2} m_3 \ddot{\theta}_2 \times \mathbf{l}_2$

of the whole system. When we are operating in the coordinate system of the UAV, the position of the center of mass $\mathbf{r_c}$ is

$$\mathbf{r_c} = \frac{1}{M} \sum_{n=1}^{3} m_n \mathbf{r_n}, \tag{9}$$

where M is the total mass of the UAV including the manipulator. $\mathbf{r_n}$ are position vectors of centers of mass of individual links of the manipulator. Forces described above need to be decomposed to their tangential and normal part due to the pivot point. We can vector sum forces with same point of application to simplify the problem as

$$\mathbf{F_i} = \sum_{n=1}^{2} \mathbf{F}_{Cin} + \mathbf{F}_{Ei2}, \tag{10}$$

where $i \in \{2, 3\}$. The tangential and normal vectors can be described as

$$\begin{aligned} \mathbf{u_i^n} &= \mathbf{c_i} - \mathbf{r_c}, \\ \mathbf{u_i^t} &= (\mathbf{u_i^n} \times \mathbf{r_2^{(B)}}) \times \mathbf{u_i^n}, \end{aligned} \tag{11}$$

where $\mathbf{c_i}$ is the position of the center of mass of the i^{th} manipulator's link and $r_2^{(B)}$ is vector $\mathbf{r_2}$ from (7) in B coordinate system. The tangential and normal force can be calculated as

$$\begin{aligned} \mathbf{F_i^n} &= p_i \mathbf{u_i^n}, \\ \mathbf{F_i^t} &= q_i \mathbf{u_i^t}. \end{aligned} \tag{12}$$

Parameters p_i and q_i can be obtained by solving following equation (e.g. by using backslash operator in Matlab):

$$\mathbf{F_i}^T = (\mathbf{u_i^{n}}^T \mathbf{u_i^{t}}^T) \begin{pmatrix} p_i \\ q_i \end{pmatrix}. \tag{13}$$

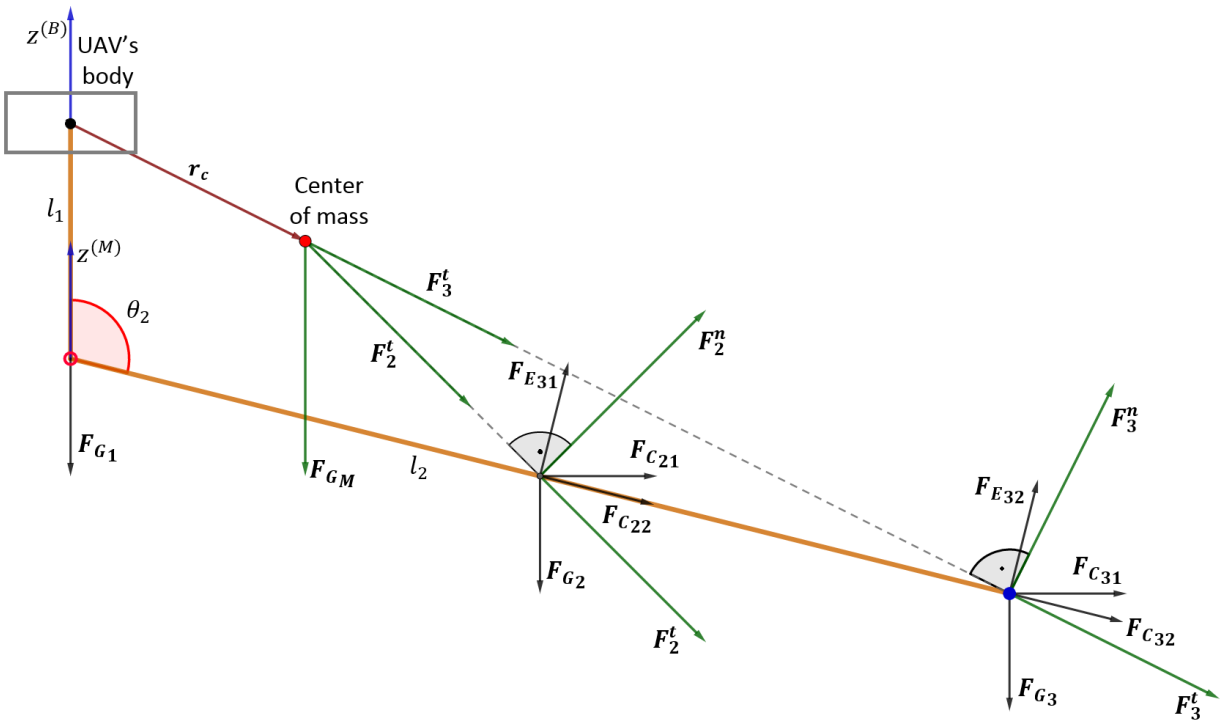


Figure 5: Principle of a force analysis. Decomposed forces $\mathbf{F_i^t}$ are related to the new point of application (center of mass).

The resulting force $\mathbf{F_R}$ is obtained by the vector sum of normal part of forces

$$\mathbf{F_R} = \sum_{i=2}^{3} \mathbf{F}_i^n + \sum_{i=1}^{3} \mathbf{F_{G}}_i. \tag{14}$$

The point of application of this force is located in the center of mass $\mathbf{r_c}$. The resulting moment is computed similarly the resulting force by adding individual moments related to the pivot point. The final torque is obtained as

$$\mathbf{M_R} = \sum_{i=2}^{3} \mathbf{r}_i \times (\mathbf{F}_i^t + \mathbf{F}_{Ei1}), \tag{15}$$

where the moment arm $\mathbf{r}_i$ is the position vector $\mathbf{u_i^n}$. Principle of force analysis is shown in figure 5. There is plane given by the first link and the second link. Thus the Euler forces with origin in first joint motions ($\mathbf{F}_{E21}, \mathbf{F}_{E31}$) are not displayed.

5 Modeling and assembling of the manipulator

As mentioned in the introduction, the manipulator was modeled in the realistic Gazebo simulator as well as assembled as a hardware prototype. The model was made to provide the option to identify the system. The hardware prototype was assembled for a purpose of testing actuators and communication with the system.

5.1 Simulating forces acting on the UAV

As the first approximation, effect of the manipulator were simulated as forces acting on the UAV in Gazebo simulator. These forces, introduced in section 4, were added to the simulator by calling the service ApplyBodyWrench[10]. This service allows us to act with defined force or torque on body of Gazebo model. It was the body of the UAV for our purpose. Parameters, which were necessary to the service were:

- Reference point - location of the point of application of acting forces
- Wrench - magnitudes and directions of calculated forces and torques
- Duration - duration of wrench application time. Forces were calculated with rate of 50 Hz, so the duration was set proportional to this value ($T_{dur} = \frac{1}{50}$ s).

This approximation of the manipulator should be enough accurate for this task. However poor observation of actions during manipulator's movements (joint states were printed in the linux terminal) bring us to simulate the manipulator by the Universal Robot Description Format, where the manipulator is visualized.

5.2 Modeling for Gazebo simulator

It is necessary to do several unavoidable steps to create a functional and precise model for Gazebo simulator. This thesis provides basic description of the process of creating a model. For detailed instructions it is adviced to follow the official tutorial[11].

5.2.1 The URDF

The Universal Robotic Description Format (URDF) is an XML file format which contains the description for the robots kinematics and dynamics in ROS. First, we have

[10] `http://docs.ros.org/jade/api/gazebo_msgs/html/srv/ApplyBodyWrench.html`
[11] `http://gazebosim.org/tutorials/?tut=ros_control`

to define the kinematic structure of the manipulator. There are two main tags for this purpose - the link tag and the joint tag. Link tags describe the shape and the position of the individual links in the easiest variant of description. Basic shapes are cylinders, spheres and boxes for the element link.

```
<link name="base1">
  <inertial>
    <mass value="0.125" />
    <inertia ixx="0.00000312" ixy="0.0" ixz="0.0" iyy="0.00000912" iyz="
   0.0" izz="0.00000512"/>
  </inertial>
  <collision>
    <geometry>
      <box size="0.05 0.04 0.004" />
    </geometry>
  </collision>
  <visual>
    <geometry>
      <box size="0.05 0.04 0.004" />
    </geometry>
  </visual>
</link>
```

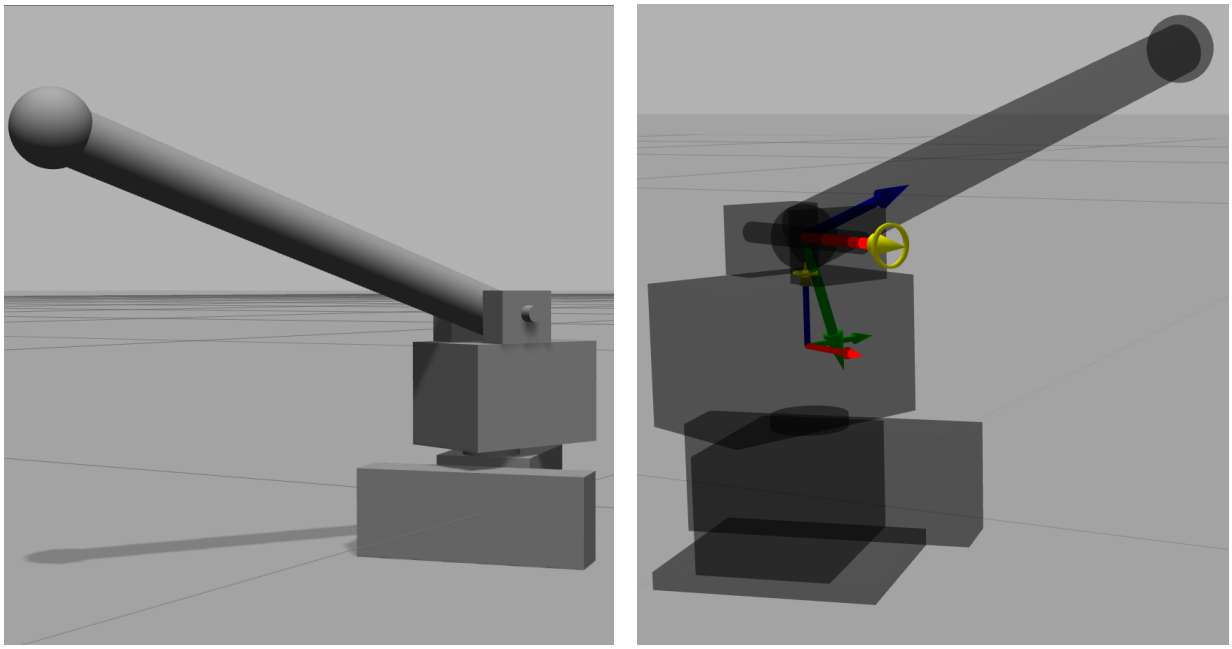
Listing 1: Description of the base link.

The joint tag allows to create linkage between two links. For this task, we are using a fixed and revolute joint types. For the revolute joint it is necessary to add limits of this joint, where limits of the lower and upper joint is needed to be determined for example.

```
<joint name="first_joint" type="revolute">
  <parent link="base1"/>
  <child link="second_link"/>
  <axis xyz="0 0 1"/>
  <limit upper="3.1415" lower="-3.1415" effort="5" velocity="2" />
  <origin xyz="0.0 0.011 0.047" rpy="0.0 0.0 0.0"/>
</joint>
```

Listing 2: First joint description. This joint is joint of revolute type.

The next step is extension of the link tags with collision properties. Collision properties define a space which can collide with other links of the same attribute. The collision can be define with the same geometry definition as in case of visual description.



(a) Front side. (b) Back side with transparent view.

Figure 6: Modeled manipulator.

The last and needful thing is defining of dynamic properties of the manipulator in the inertia XML tag. We need to set the mass of each link and the 3×3 inertia matrix. Since the matrix is symmetrical, it is represented by only 6 independent parameters. The inertia tensor depends both on the mass and the distribution of mass of the object. Due to the fact, that the designed manipulator consists of simple shapes, the inertia tensor parameters can be found in literature[12]. However, we will deal with inertia tensor more in the next chapter. Finally, tags that describe color of links, were added to the URDF file. It is just detail, but individual links can be then recognized easily. Image of the model is in fig. 6. In the fig. 6b, there is a transparent view, where coordinate systems of individual revolute joints are visible. Axis with yellow hat signaling the axis of rotation of the joint.

5.2.2 Transmission elements and the control plugin

To be able to control our robot in the Gazebo simulator, we have to add transmission elements to a URDF. These transmission elements are used to link actuators to joints. There are several tags prepared for transmission, however only one configuration is currently[13] implemented.

[12] `https://en.wikipedia.org/wiki/List_of_moments_of_inertia`
[13] `http://wiki.ros.org/urdf/XML/Transmissiona`

```
<transmission name="first_joint">
  <type>transmission_interface/SimpleTransmission</type>
  <actuator name="motor1">
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
  <joint name="first_joint">
    <hardwareInterface>EffortJointInterface</hardwareInterface>
  </joint>
</transmission>
```

Listing 3: Example of transmission XML tag

Additionally, the gazebo_ros_control plugin needs to be added to parse the transmission tags and to load the appropriate hardware interfaces and the controller manager.

```
<gazebo>
  <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
    <robotNamespace>/2dof_model</robotNamespace>
  </plugin>
</gazebo>
```

Listing 4: Adding gazebo control plugin

5.2.3 Configuration file and launch file

Settings for controller of each joint need to be located in configuration file. This file is in the same package as our URDF file. In configuration file we can set the PID controller gains and type of controller.

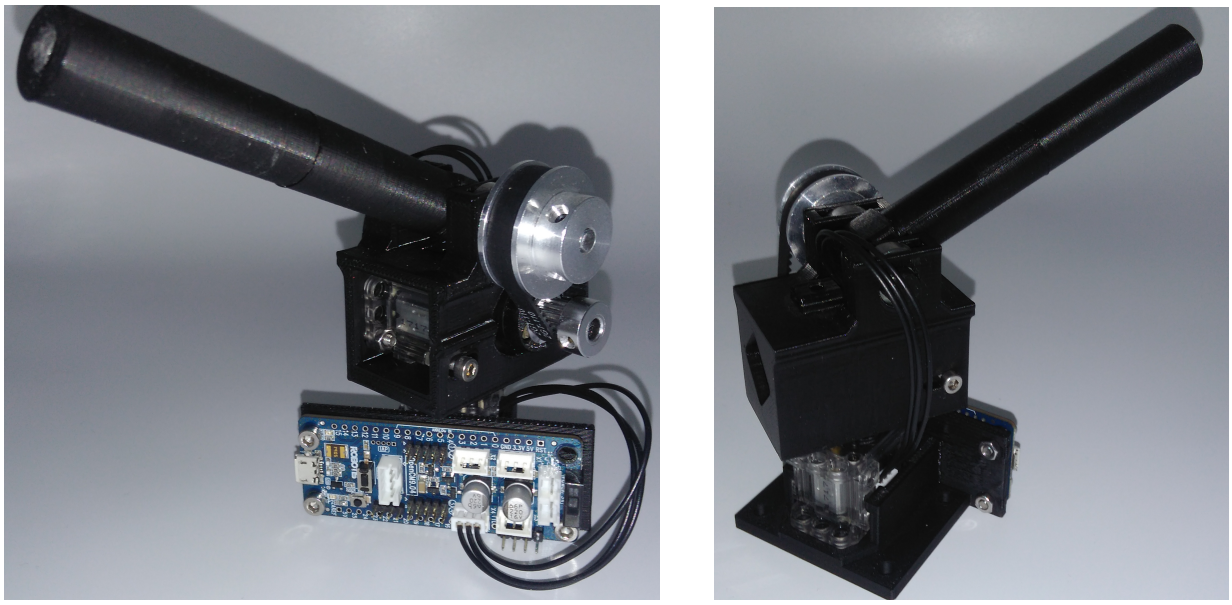
```
2dof_model:
    joint_state_controller:
        type: joint_state_controller/JointStateController
        publish_rate: 50

    first_joint_controller:
        type: effort_controllers/JointPositionController
        joint: first_joint
        pid: {p: 0.1, i: 0.02, d: 0.03}
```

Listing 5: Example of controller setting in configuration file

Gains of the PID controller were debugged by dynamic reconfiguration[14] in ROS. The launch file was created in the same package then. It loads joint controller configurations and loads all of the defined controllers. When our URDF model is spawned in Gazebo and the prepared launch file is launched, we can control joints of the manipulator by publishing ROS messages.

5.3 Hardware assembling



(a) Front side. Gearbox and control board is visible. (b) Back side.

Figure 7: Assembled hardware prototype of the manipulator.

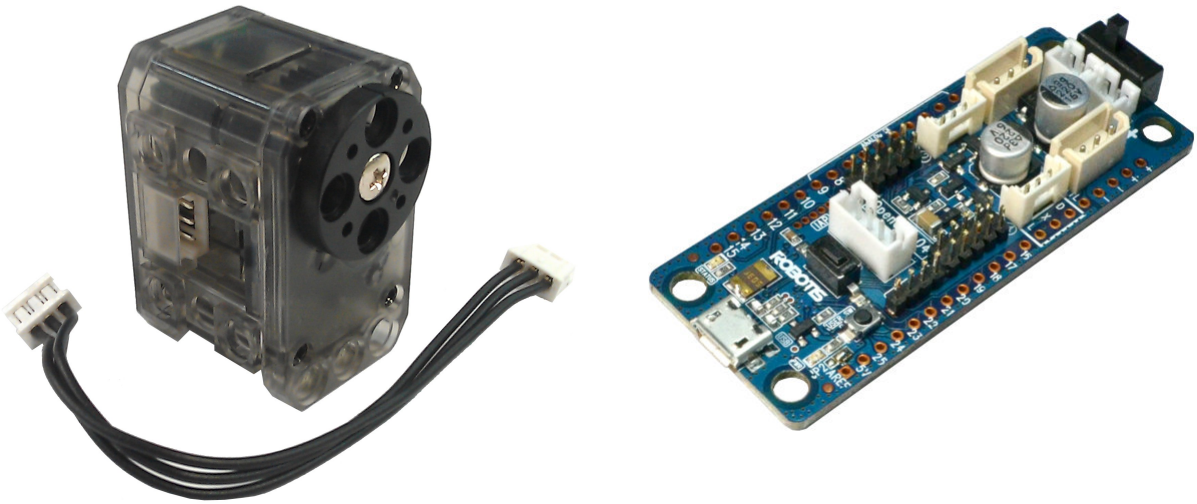
Servomotors Dynamixel XL-320 [16] (fig. 8a) were chosen for the hardware prototype and OpenCM 9.04c [17] (fig. 8b) was chosen as the control board. Communication between the UAV and the manipulator is implemented via serial communication. Joint coordinates, calculated in the UAV control board, are sent through this communication channel. The servomotors are controlled on the bus, thus it is necessary to set the unique ID to each motor. Design of the prototype was modeled so that it has the same attributes as the Gazebo model. Due to the fact, that used servomotors have low torque, their range is excessive to their usage (up to 300 degrees) and the second joint have to handle higher loads than the first joint (it overcomes the gravity force acting in the end-effector), the second joint was equipped with a gearbox. It consists of two aluminum pulleys and a ring belt with conversion ratio 3:8. The shaft of the third link is fixed in two radial ball bearings. Links of the manipulator were printed on a 3D printer due to requirements on

[14] `http://wiki.ros.org/dynamic_reconfigure`

low weight. Models for printing were created using Fusion360 software. The final form of the manipulator is shown in fig. 7.

5.3.1 Servomotor and control board

A servomotor is a rotary actuator whose position is regulated by closed-loop control. It consist of DC electric motor, gearbox and control electronics. The necessary part of electronics is a rotary encoder that works as sensor of angular position. The signal from the encoder is brought to input of control electronics, where the signal is converted and compared with the setpoint of the control board. There are several methods of comparing described input signals. In common servomotors, there is a monostable multivibrator circuit. It generates negative signal corresponding to the angular position. This signal is added to the positive control signal, which represents required angular position. The differential signal is amplified and brought to the DC electric motor.



(a) Dynamixel XL-320 [15]. (b) OpenCM-9.04-C [21].

Figure 8: Manipulator's electronic parts.

The servomotor, which was used in this thesis, works on the similar principle, that was described above. Moreover it uses its dynamical model in the open-loop and PID controller as the main control method. Dynamixel XL-320 has many parameters, as velocity and load feedback, communication via bus, etc. This makes it useful actuator in robotics applications. On the other hand, its mechanical construction has flaws. Even the gearbox and the main shaft are plastic, thus the servomotor is not very stiff and motions with load are inaccurate. Another consequence is, that the manipulator is gently bending around the

first joint and the pulley on the second joint can not be fully tightened. It follows, that Dynamixel XL-320 is an actuator suitable for undemanding robotic operations. For the next research it will be necessary to rebuild the manipulator with new and more robust actuators.

To control the servomotors, the OpenCM 9.04c control board was chosen. It is a microcontroller board based on 32bit ARM Cortex-M3. Board's schematic and source code are open source. For development and programming of the manipulator's electronics we use the OpenCM ROBOTIS integrated development environment (IDE), which connects to the microcontroller via USB (universal serial bus). Libraries of this application directly support work with Dynamixel servomotors. From the above described follows, that structure and work with OpenCM boards is similar to Arduino[15]-type boards.

[15] `https://www.arduino.cc/`

6 UAV model

In this section, the UAV dynamic model will be introduced. Together with the UAV model, the manipulator model will be introduced as well. In the end, the manipulator model from section 5.2.1 will be attached to the UAV's Gazebo model.

6.1 The UAV dynamic model

The UAV dynamic model is described in this chapter. The hexacopter was used for experiments, however, without less of generality, we work with a model of a quadrotor, since we do not directly interact with thrust of individual motors, which is handled by an integrated UAV stabilization board. We introduce two coordinates systems. The first one is the world coordinate system W. Its position is fixed to specific place in the world. We use ENU convention here, thus the axes point to the east, north and upward. The second coordinate system B is the UAV body system. This coordinate system was indicated in chapter 2. It is fixed to the UAV's body and its origin is in the center of mass of the quadcopter. The orientation of individual axes are illustrated in figure 9. The dynamic model [6] can be represented by following equations

$$\dot{\mathbf{x}} = \mathbf{v}, \tag{16}$$

$$m\dot{\mathbf{v}} = m\mathbf{g} - \mathbf{R}\mathbf{T}, \tag{17}$$

$$\dot{\mathbf{R}} = \mathbf{R}\widehat{\Omega}, \tag{18}$$

$$\mathbf{M} = \mathbf{J}\dot{\Omega} + \Omega \times \mathbf{J}\Omega, \tag{19}$$

where

- $\mathbf{x}$ the position of the center of mass in the world coordinates system
- $\mathbf{g}$ the gravity acceleration $\begin{pmatrix}0 & 0 & -g\end{pmatrix}^T$
- $\mathbf{R}$ the rotation matrix from the UAV coordinate system to the world coordinate system
- $\mathbf{T}$ the total thrust generated by propellers $\begin{pmatrix}0 & 0 & T\end{pmatrix}^T$
- $\mathbf{M}$ the total moment acting on the UAV in its coordinate system
- $\mathbf{J}$ the inertia matrix of the quadorcopter
- Ω the angular velocity of the UAV
- $\widehat{\Omega}$ the map $\widehat{\ }: \mathbb{R}^3 \to SO(3)$ such that $x \times y = \widehat{x}y$, for all $x, y \in \mathbb{R}^3$
- m the total mass of the quadcopter

The total thrust can be computed as sum of thrusts of individual propellers. The

total moment is sum of moments related to individual axes, specifically

$$\begin{aligned}\mathbf{M}_x &= (-rT_1 + rT_3)\mathbf{e_x},\\ \mathbf{M}_y &= (-rT_2 + rT_4)\mathbf{e_y},\\ \mathbf{M}_z &= (-rc(T_1 + T_3) + rc(T_2 + T_4))\mathbf{e_z},\end{aligned} \tag{20}$$

where T_i is thrust of i^{th} propeller, $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are unit vectors with direction of individual axes and r is distance of propellers axis of rotations from B coordinate system origin. The coefficient c expresses transfer from thrust to force effecting in xy plane.

6.2 Model extension

In this section, we modify the dynamic model (16-19) presented in chapter 6.1. Our goal is creation of new equations, that describe the UAV with the manipulator attached. The coordinate system C of the combined system is similar to the previous one. Its origin is located in the center of mass of the extended model. The axis are parallel to their original version. Modification of the first equation (16) can be written as

$$\dot{\mathbf{x}}_N = \mathbf{v}_N, \tag{21}$$

where $\dot{\mathbf{x}}_N$ is the position of the new center of mass in the world coordinate system. However, if manipulator's joints are in motion, this position is floating even when the UAV is stabilized. Thus, if we need to track position of the UAV, we can obtain it from previous relation as

$$\dot{\mathbf{x}}_{UAV} = \dot{\mathbf{x}}_N - \mathbf{R}\mathbf{r}_c, \tag{22}$$

where $\mathbf{r}_c$ is position of the center of mass in the B coordinate system described in section 4.4. The second equation (17) will be in following form:

$$(m_{UAV} + m_M)\dot{\mathbf{v}}_N = m_{UAV}\mathbf{g} + \mathbf{F}_R - \mathbf{R}\mathbf{T}, \tag{23}$$

where m_M is mass of the manipulator and $\mathbf{F}_R$ is total force, with originates in the manipulator's motions. This force was computed in section 4.4. We can notice, that mass of the manipulator is missing on the right side of equation with gravitational acceleration. The reason is that force of gravity acting on the manipulator is part of the force $\mathbf{F}_R$.

The third equation (18) will remain the same in this case. This relationship represents change of the rotation matrix $\mathbf{R}$ in time. Thus it describes change in transformation between two coordinate systems independently on the system itself.

The last equation (19) is modified as

$$\mathbf{M}_N + \mathbf{M}_R = \mathbf{J}_N\dot{\Omega} + \Omega \times \mathbf{J}_N\Omega, \tag{24}$$

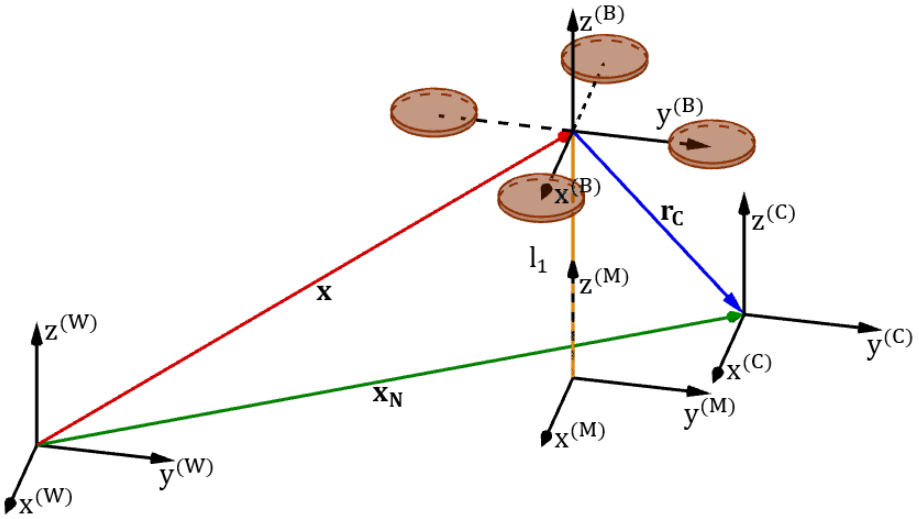


Figure 9: All introduced coordinate systems.

where $\mathbf{M}_R$ is the moment described in section 4.4. The new moment $\mathbf{M}_N$, with source in thrust of individual propellers, is extension of equations (21) and it will be in following form:

$$\mathbf{M}_N = \sum_{n=1}^{4} \mathbf{r}_i \times \mathbf{T}_i + \sum_{n=1}^{2} cr_{2i} T_{2i} \mathbf{e}_z - \sum_{n=1}^{2} cr_{2i-1} T_{2i-1} \mathbf{e}_z, \tag{25}$$

where $\mathbf{r}_i$ is position of individual propellers in the C coordinate system and $\mathbf{e}_z$ is the unit vector with direction of z-axis.

6.2.1 Tensor of inertia

The inertia tensor (matrix) consists of inertia and products of inertia about three coordinate axes. It depends on distribution of the mass in body and choice of the coordinate system. The inertia matrix $\mathbf{J}$ for a quadrocopter is a diagonal matrix, because quadrocopters are symmetrical when coordinate system B is used. This matrix is already known and we can write it as

$$\mathbf{J}_q^{(B)} = \begin{pmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{pmatrix}. \tag{26}$$

Another inertia tensor, that is necessary to describe, is the inertia tensor of the manipulator. We assume, that mass is distributed equally in the individual links. Without loosing much of an accuracy, we may consider the base link and the first link of the manipulator as one link. At first, we define simple objects, which form the resulting shape of the manipulator.

- A solid cuboid (box) represents the first manipulator's link with inertia tensor $\mathbf{J}_b$. Its parameters are width w_b, height h_b, depth d_b and mass $m_b = m_1$.
- A solid cylinder represents the second manipulator's link with inertia tensor $\mathbf{J}_c$. Its parameters are radius r_c, height h_c and mass $m_c = m_2$.
- A solid sphere represents the weight located on the position of the end-effector with inertia tensor $\mathbf{J}_s$. Its parameters are radius r_s, mass $m_s = m_2$.

These inertia tensors are expressed in the M coordinate system as

$$\mathbf{J}_b^{(M)} = \mathbf{J}_b + \mathbf{J}(0, 0, -\frac{h_b}{2}), \tag{27}$$

$$\mathbf{J}_c^{(M)} = \mathbf{T}^T(\mathbf{J}_c + \mathbf{J}(0, 0, \frac{h_c}{2}))\mathbf{T}, \tag{28}$$

$$\mathbf{J}_s^{(M)} = \mathbf{J}_s + \mathbf{J}(-x_E^{(M)}, -y_E^{(M)}, -z_E^{(M)}), \tag{29}$$

where $\mathbf{J}(x, y, z)$ is application of Steiner's theorem [13] with the meaning of (30).

$$\mathbf{J}(x, y, z) = m \begin{pmatrix} y^2 + z^2 & -xy & -xz \\ -xy & z^2 + x^2 & -yz \\ -xz & -yz & x^2 + y^2 \end{pmatrix}, \tag{30}$$

where m is the total mass of the body. $\mathbf{T}$ is a matrix of transformation in the form

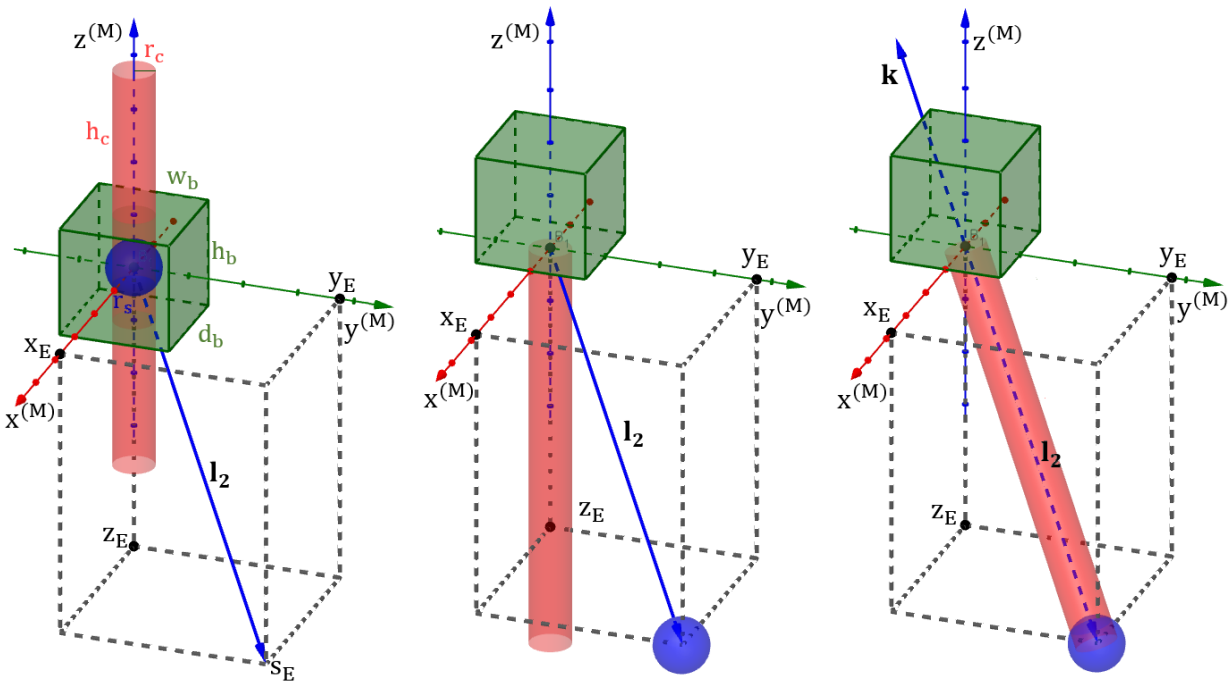
$$\mathbf{T} = \begin{pmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \end{pmatrix}, \tag{31}$$

where $\mathbf{i}, \mathbf{j}, \mathbf{k}$ are unit vectors with directions of axes of rotated coordinate system M. For our purpose, the most important is the vector $\mathbf{k}$, which have direction of $-\mathbf{l}_2$ (7). The inertia tensor of the manipulator can be now written as

$$\mathbf{J}_m^{(M)} = \mathbf{J}_b^{(M)} + \mathbf{J}_c^{(M)} + \mathbf{J}_s^{(M)}. \tag{32}$$

Meaning of individual inertia tensors and operations with them, which were described above is indicated in the figure 10 in the coordinate system M. In the figure 10a, there is representation of object with inertia tensors of simple shapes. In the next step, simple shapes are translated from the coordinate system origin to their right positions (fig 10b). In the end, rotation is applied on the second link (fig 10c).

The inertia matrix $\mathbf{J}_m^{(M)}$ is variable and depends on actual states of manipulator's joints. The inertia tensors described above were expressed in different coordinate system. However, they all have to be expressed in the coordinate system C. Steiner's theorem can be used here, thanks to the fact, that coordinate systems M, B and C can be formed by



(a) $\mathbf{J}_b, \mathbf{J}_c, \mathbf{J}_s$. (b) $\mathbf{J}_b^{(M)}, \mathbf{J}_s^{(M)}, \mathbf{J}_c + \mathbf{J}(0, 0, \frac{h_c}{2})$. (c) $\mathbf{J}_b^{(M)}, \mathbf{J}_c^{(M)}, \mathbf{J}_s^{(M)}$.

Figure 10: Meaning of inertia tensors. A visualized approximation of manipulator's inertia tensor is in fig. 10c.

translation of one of them (thus without rotation). Using the theorem, the inertia tensors in C coordinate system are

$$\mathbf{J}_m^{(C)} = \mathbf{J}_m^{(M)} + \mathbf{J}_M^C, \tag{33}$$
$$\mathbf{J}_q^{(C)} = \mathbf{J}_q^{(B)} + \mathbf{J}_B^C, \tag{34}$$

where tensors $\mathbf{J}_M^C, \mathbf{J}_B^C$ are in following form:

$$\mathbf{J}_G^H = \mathbf{J}(x, y, z), \tag{35}$$

where $\mathbf{J}(x, y, z)$ is meaning of Steiner's theorem from (30), G is the original coordinate system, H is the translated coordinate system, m is the total mass of the body and $\mathbf{r} = (x, y, z)$ is vector of translation from G to H. Finally, the inertia tensor of combined system is

$$\mathbf{J}_N = \mathbf{J}_m^{(C)} + \mathbf{J}_q^{(C)}. \tag{36}$$

6.3 State-space model of the disturbance

Discrete time-invariant linear state-space model was used for modeling of disturbance of the UAV. It uses state variables to describe a physical system by a set of first-order difference equations. An unknown model parameters can be determined experimentally when input-output data are known. This will be used for identification of our model in the next chapter. Representation of a model with described attributes can be written in the following form:

$$\mathbf{q}_{(k+1)} = \mathbf{A}\mathbf{q}_{(k)} + \mathbf{B}\mathbf{u}_{(k)}, \tag{37}$$

$$\mathbf{r}_{(k)} = \mathbf{C}\mathbf{q}_{(k)} + \mathbf{D}\mathbf{u}_{(k)}, \tag{38}$$

where $\mathbf{q} \in \mathbb{R}^{n\times 1}$ is the state vector, $\mathbf{u} \in \mathbb{R}^{m\times 1}$ is the input vector, $\mathbf{r} \in \mathbb{R}^{p\times 1}$ is the output vector, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the state matrix, $\mathbf{B} \in \mathbb{R}^{n\times m}$ is the input matrix, $\mathbf{C} \in \mathbb{R}^{p\times n}$ is the output matrix, $\mathbf{D} \in \mathbb{R}^{p\times m}$ is the feedforward matrix and k is the sample time. The second equation will not be used further as we need to operate with physical quantities that are state variables directly. State-space model of deflection in x-axis has following form:

$$\mathbf{x}_{(k+1)} = \begin{pmatrix} 1 & t_s & 0 \\ 0 & 1 & t_s \\ 0 & 0 & p_1 \end{pmatrix} \mathbf{x}_{(k)} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ p_2 & p_3 & p_4 \end{pmatrix} \mathbf{u}_{x(k)}, \tag{39}$$

where state vector $\mathbf{x} = \begin{pmatrix} x & \dot{x} & \ddot{x} \end{pmatrix}^T$, t_s is sampling period (set as 0.02 s in this thesis), and $p_{1..4}$ are parameters of the first order transfer function. The input vector was selected as

$$\mathbf{u}_x = \begin{pmatrix} \Delta\dot{x} & F_x & M_y \end{pmatrix}^T, \tag{40}$$

where F_x and M_y are components of vectors of the total force and moment from section 4.4 of this thesis and $\Delta\dot{x}$ is difference of actual velocity from desired velocity. The first input represents behavior of integral components of the UAV's regulators and allows initial oscillations of the model given to the system in Gazebo simulator. Due to the fact, that quadcopter is symmetrical vehicle, the state-space model of disturbance in y-axis is identical. Thus, that model has the state vector $\mathbf{y} = \begin{pmatrix} y & \dot{y} & \ddot{y} \end{pmatrix}^T$, the state matrix $\mathbf{A}_y = \mathbf{A}_x$, the input matrix $\mathbf{B}_y = \mathbf{B}_x$ and input vector $\mathbf{u}_y = \begin{pmatrix} \Delta\dot{y} & F_y & M_x \end{pmatrix}^T$. The model of the disturbance in altitude is not described. The reason is mentioned and analyzed in section 7. The last system represents the disturbance in yaw. Equation of this system is written as

$$\boldsymbol{\phi}_{(k+1)} = \begin{pmatrix} 1 & t_s \\ 0 & p_5 \end{pmatrix} \boldsymbol{\phi}_{(k)} + \begin{pmatrix} 0 & 0 \\ p_6 & p_7 \end{pmatrix} \mathbf{u}_{\phi(k)}, \tag{41}$$

where state vector $\boldsymbol{\phi} = \begin{pmatrix} \phi & \dot{\phi} \end{pmatrix}^T$, $p_{5..7}$ are parameters of the first order transfer function and $\mathbf{u}_\phi = \begin{pmatrix} \Delta\phi & M_z \end{pmatrix}^T$. $\Delta\phi$ has the same meaning as $\Delta\dot{x}$ and $\Delta\dot{y}$ described above. M_z is component of total moment acting on the UAV around z-axis. In this thesis, identification and control is going on the UAV in the quiescent state with desired relative coordinates $x = y = \phi = 0$. Thus, we can write $\Delta\dot{x} = \dot{x}$, $\Delta\dot{y} = \dot{y}$ and $\Delta\phi = \phi$.

6.4 Attaching the manipulator to the Gazebo UAV model

The used hexacopter Gazebo model is formed by several Xacro files. Xacro is an XML macro language, that allows us to simplify and make URDF files clearer. It supports constants, math operations and macros. The newly created manipulator description was directly inserted into new macro in Xacro file, where the hexacopter's body is defined. In the modified Xacro file a new fixed joint connecting of the hexacopter's body and the manipulator's base link was added. It was necessary to rotate and translate the manipulator to the right position towards the quadcopter. For this action, the connecting joint's parameters (described in section 5.2.1) were changed.

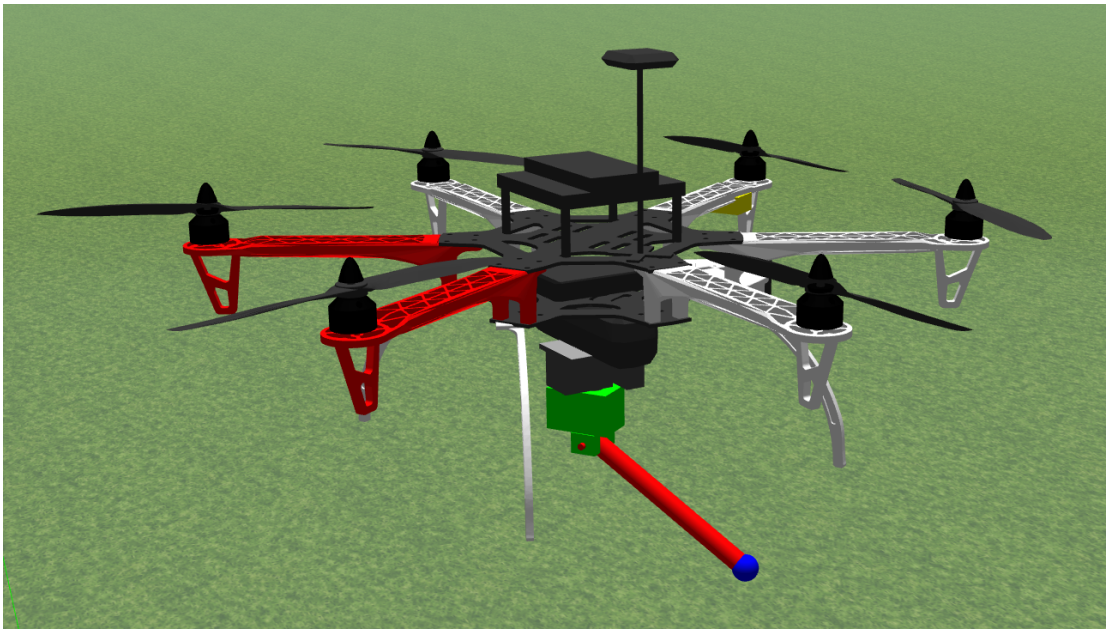


Figure 11: The UAV Gazebo model equipped with the manipulator model

The modified Gazebo model during simulated flight can be seen in fig. 11). The manipulator is fixed directly below the UAV's battery. Colors of individual links were added as mentioned in (5.2.1). The particular designation is: base link (grey), first link (green), second link (red), weight on position of the end effector (blue).

7 System identification

In this section, parameters $p_{1...7}$ are determined. Even when step change of input values can be done (see 4.1) for step response identification, mathematical optimization [10] was chosen as better way to identify parameters of system. However, step response identification is unsuitable for unstable systems as the UAV. Several simulated flights were conducted for this task. During these simulations, position states of the UAV and joint states of the manipulator were recorded with sampling period t_s. The UAV's velocity and acceleration was calculated from position values by differentiating as well as the total force and the total torque was calculated from joint states according to section 4.4. At first, we will identify the system of xy position disturbance. According to described state-space model (39), we can write the following linear equation for acceleration $\ddot{x}$

$$\ddot{x}_{(k+1)} = p_1\ddot{x}_{(k)} + p_2\Delta\dot{x}_{(k)} + p_3F_{x(k)} + p_4M_{y_{(k)}}. \tag{42}$$

To obtain wanted parameters, a large system of equations (42) has to be formulated for all measured data, typically hundreds of equations. Then we are talking about an overdetermined system of equations. These equation can be written in the matrix form

$$\begin{pmatrix} \ddot{x}_{(2)} \\ \ddot{x}_{(3)} \\ \vdots \\ \ddot{x}_{(n)} \end{pmatrix} = \begin{pmatrix} \ddot{x}_{(1)} & \dot{x}_{(1)} & F_{x1} & M_{y1} \\ \ddot{x}_{(2)} & \dot{x}_{(2)} & F_{x2} & M_{y2} \\ \vdots & \vdots & \vdots & \vdots \\ \ddot{x}_{(n-1)} & \dot{x}_{(n-1)} & F_{xn-1} & M_{yn-1} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{pmatrix}, \tag{43}$$

where n is the number of equations. Unknown parameters were obtained by least square method in Matlab by operator '\' (matrix pseudoinverse). The state matrix and the input matrix of identified system with obtained parameters and sampling period $t_s = 0.02s$ (this period was used for identification in this thesis) are then

$$\mathbf{A}_x = \mathbf{A}_y = \begin{pmatrix} 1 & 0.02 & 0 \\ 0 & 1 & 0.02 \\ 0 & 0 & 0.9718 \end{pmatrix}, \tag{44}$$

$$\mathbf{B}_x = \mathbf{B}_y = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.1876 & 0.1971 & -0.0149 \end{pmatrix}. \tag{45}$$

As we can see in figure 12, acceleration estimated by the model (42) for x-axis resembles the measured data that were used for identification with just the input $\mathbf{F}_x$ and $\mathbf{M}_y$. When we look at the figure 13, where verification of identified model on y-axis disturbances was performed, the results are not satisfactory. Even when we tried to identify the y-disturbance system separately, estimation does not produce model parameters that would allow to replicate the x-axis behavior. We do not consider, that some mistake have been done during model description. We conclude that this phenomenon originates in the simulator.

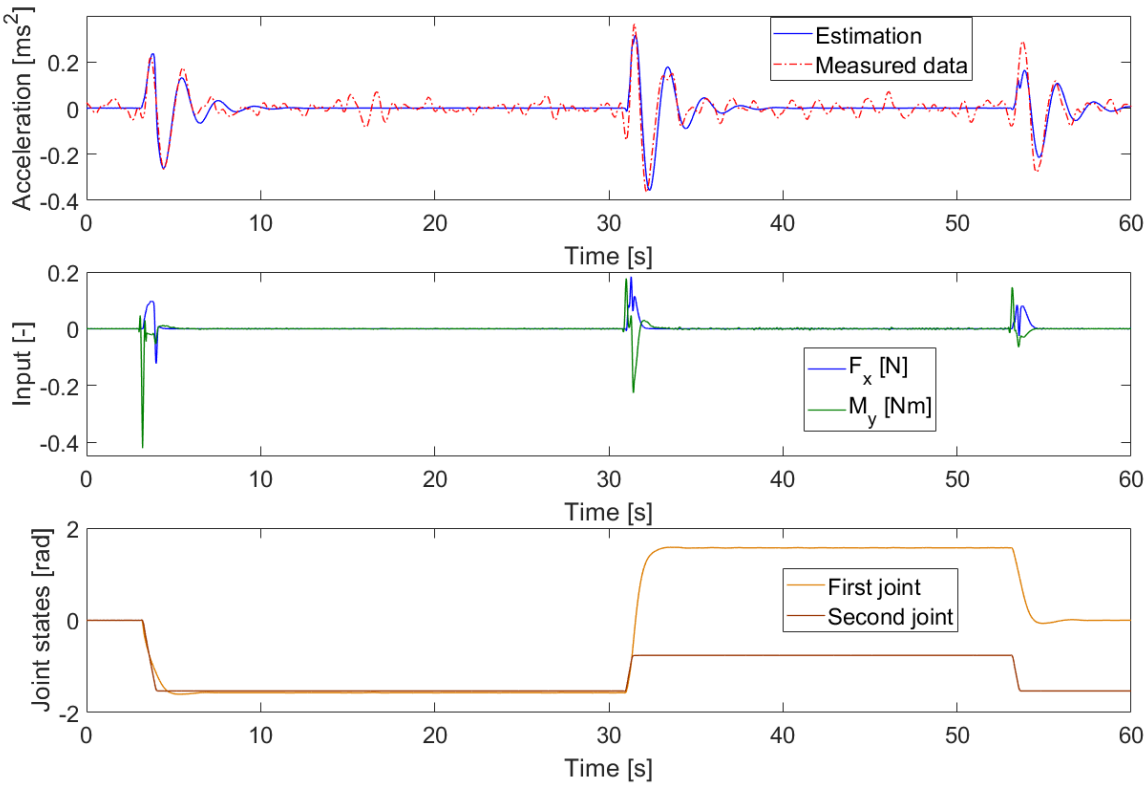


Figure 12: Identification of x-axis disturbances.

The reason of this conclusion is behavior of the UAV in the same axis while it is controlled, where the control has no effect in y-axis even when control the system is based on manually prepared data. More about this problem is in section 8.3.2. Due to the fact, that the multicopter is decoupled, we consider the system identification as successful and finding and repairing of the issue in the simulator is subject of future work.

In the next step, we will identify the system of disturbance in yaw angle. For this system, another linear equation can be written:

$$\dot{\phi}_{(k+1)} = p_5\dot{\phi}_{(k)} + p_6\Delta\phi_{(k)} + p_7 M_{z(k)}. \tag{46}$$

In the same manner as in the previous case, an overdetermined system of equations is formulated in matrix form as

$$\begin{pmatrix} \dot{\phi}_{(2)} \\ \dot{\phi}_{(3)} \\ \vdots \\ \dot{\phi}_{(n)} \end{pmatrix} = \begin{pmatrix} \dot{\phi}_{(1)} & \phi_{(1)} & M_{z1} \\ \dot{\phi}_{(2)} & \phi_{(2)} & M_{z2} \\ \vdots & \vdots & \vdots \\ \dot{\phi}_{(n-1)} & \phi_{(n-1)} & M_{zn-1} \end{pmatrix} \begin{pmatrix} p_5 \\ p_6 \\ p_7 \end{pmatrix}. \tag{47}$$

This set of linear equations was again solved by the least square method. The yaw system

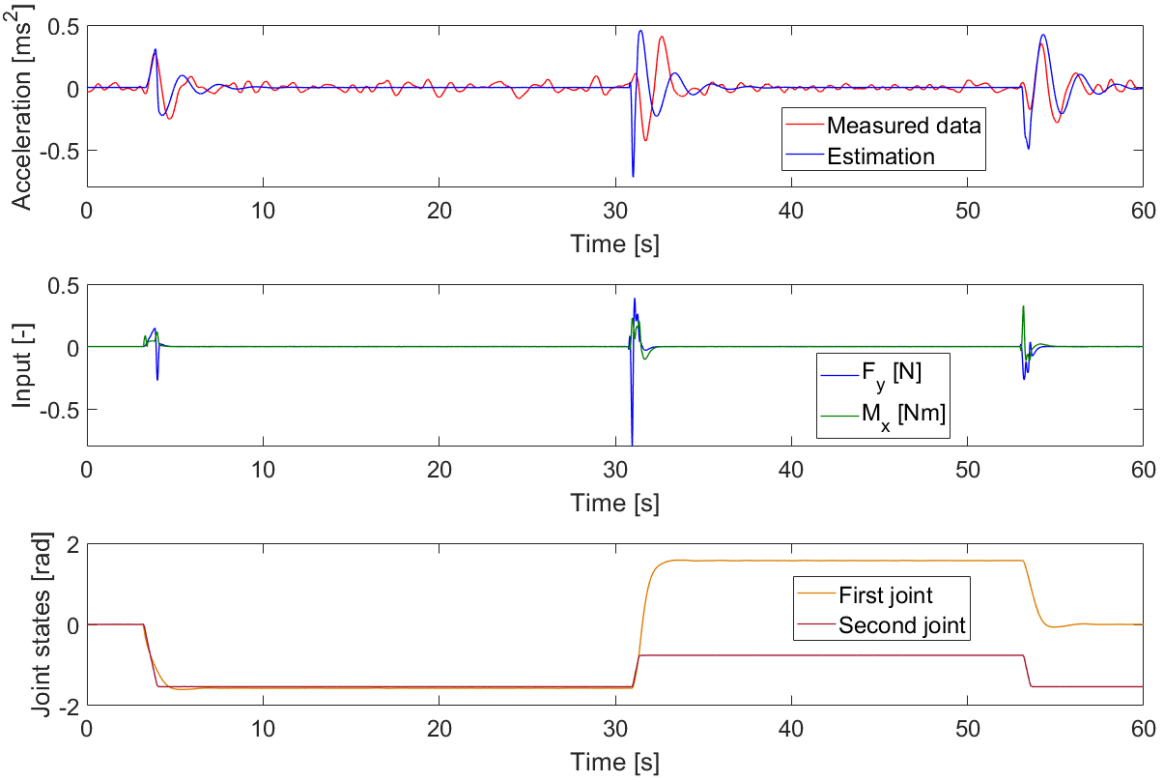


Figure 13: Estimation of y-disturbances system.

is identified described by

$$\mathbf{A}_\phi = \begin{pmatrix} 1 & 0.02 \\ 0 & 0.9091 \end{pmatrix}, \tag{48}$$

$$\mathbf{B}_\phi = \begin{pmatrix} 0 & 0 \\ -0.4686 & 0.0944 \end{pmatrix}. \tag{49}$$

Figure 14 shows verification of estimation of yaw angle. This system has been identified with high accuracy.

For the purpose of altitude identification, the measured state is the altitude. After filtration of data, velocity and acceleration was computed. It was expected that motions in vertical plane will have an impact on altitude. Therefore, motions of second joint were performed primarily during a simulated flight. When looking at the figure 15, influence of the manipulator's motions on the altitude do not significantly exceed noise. For that reason, identification of altitude has not been performed. However, influence of the whole manipulator on altitude was perceptible. Long settling time in order of tens seconds was detected while the UAV was trying to get to the desired altitude.

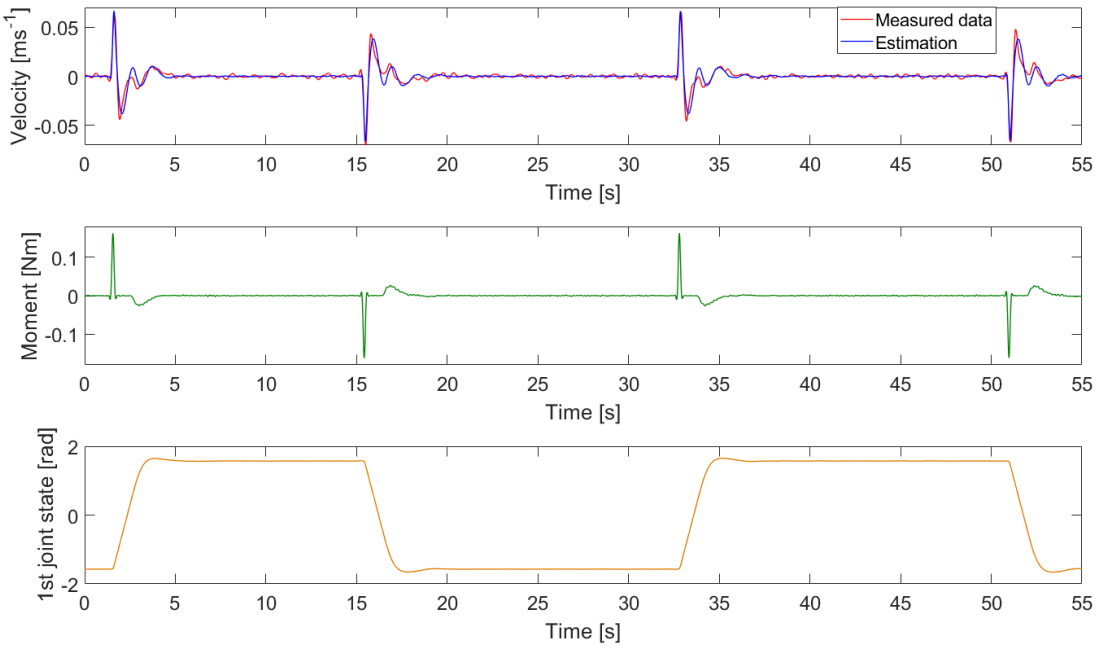


Figure 14: Identification of yaw-disturbance system.

7.1 Filtering noisy data

Measured flight data from the Gazebo simulator are necessary to identify system behavior. However these data are noisy and unfit to raw processing. A digital filter can be applied to increase signal-to-noise ratio (SNR or S/N) with low impact on the signal distortion. In this thesis, the Savitzky-Golay [19] filter was used to filter the measured data. We obtain the smoothed data output by sampling the fitted polynomial at each data point. The result is the same if we do a combination of the sub-set of adjacent data points with weighted coefficients. These coefficients can be computed for given polynomial order and length of sub-set data. The output samples of S-G filter can be computed by a discrete convolution of the form

$$x_i^S = \sum_{j=-m}^{m} c_j x_{i+j}^N, \tag{50}$$

where x_i^S are smoothed data, x_{i+j}^N are noisy data, c_j are convolution coefficients and m can be set as

$$m = \frac{n-1}{2}, \tag{51}$$

where n is a length of sub-set of adjacent data points. The simplest and the least effective variant of S-G filter is Moving average filter, where

$$c_j = \frac{1}{n}. \tag{52}$$

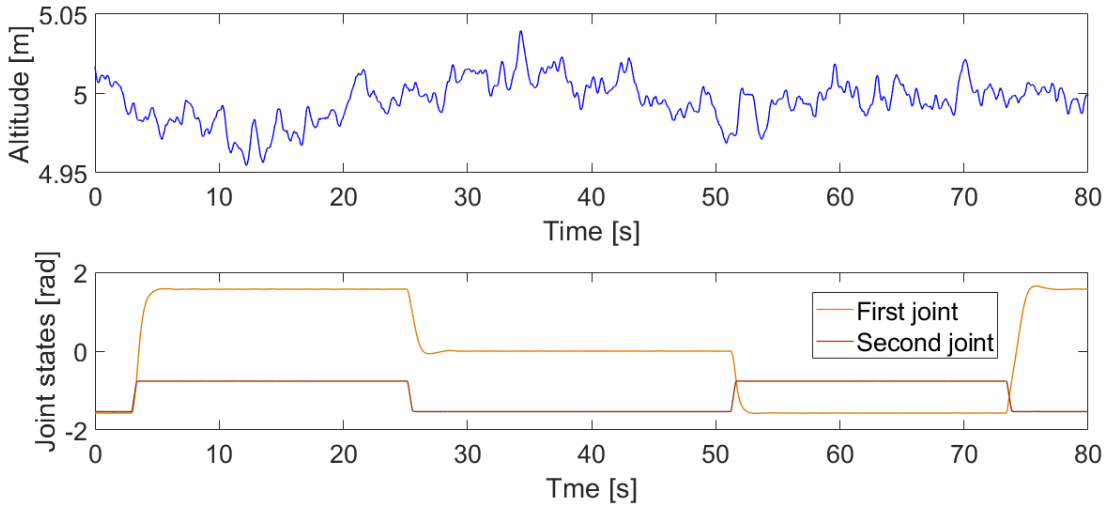


Figure 15: Influence of joints motions on the altitude. It is obvious that disturbances in altitude do not exceed a noise.

In this thesis, the S-G filter was used through Matlab where the filter is implemented in function *smooth*. An example of its use is shown in figure 16.

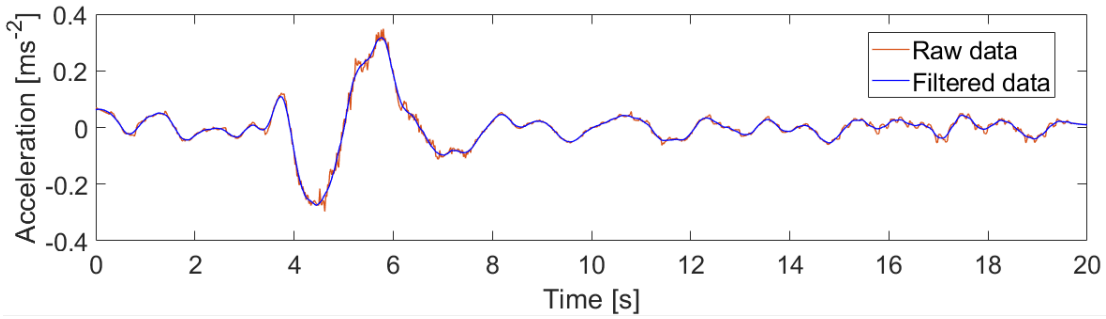


Figure 16: Acceleration in x-axis filtered with Savitzky-Golay filter.

8 Control of the UAV

In this section, we propose a system for mitigation of identified disturbances. Two methods of control were designed for that purpose. These two methods will be compared and we will find out which method is preferable for this task.

8.1 Control pipeline

Several control layers are implemented in the used control system of the UAV. Figure 17 shows the simplified structure of the control system for the single UAV, which will be described below. More detailed description can be found in [1]. On the lowest level, there is the onboard attitude controller. The PixHawk flight controllers are used on the UAVs in the MRS laboratory. The input of this controller is desired orientation and total thrust and the output is desired motor speed. The non-linear SO(3) controller [6] is in the second layer. It uses the model (6.2). The input of this controller is position and yaw angle and their derivations. On the top of the control system, there is the MPC trajectory tracker [1]. It accepts desired trajectory of the UAV.

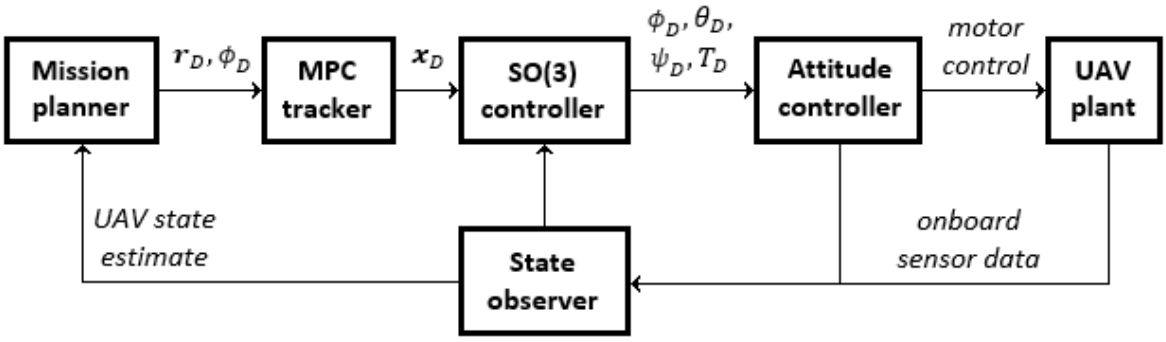


Figure 17: Control pipeline diagram showing the components responsible for control and guidance of the UAV.

8.2 Control by mirrored trajectory

At first, we designed a method of controlling deflected UAV based on error trajectory identification. This trajectory was estimated thanks identified model (42). For example,

points of trajectory in x-axis x_n are estimated as

$$\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix}_{(k+1)} = \mathbf{A}_x \begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix}_{(k)} + \mathbf{B}_x \mathbf{u}_x, \tag{53}$$

with initial condition

$$\begin{pmatrix} x \\ \dot{x} \\ \ddot{x} \end{pmatrix}_{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{54}$$

Input vector $\mathbf{u}_x$ has the same meaning as the input vector in case of identification. Thus, it consists of forces and torques acting on the UAV. Estimated trajectory was mirrored and dispatched to the MPC tracker. The error trajectory and its mirrored counterpart should be added and reduce error with origin in the manipulator's motions.

8.2.1 Summary

Experiments based on control of the UAV by mirrored trajectory did not lead to satisfactory results. At first we tried to pass the predicted mirrored trajectory from identified system to the MPC tracker. Even when we just filtered and mirrored previously measured trajectory, the position error created by manipulator's motions remained unchanged. The reason of this behavior is following. It may seem that the MPC tracker was developed with such parameters, so that it precisely follows trajectories of higher distances and in comparison with this, the deflection, which we tried to minimize, is so insignificant. The MPC tracker minimizes position error and control effort simultaneously. Due to the small deflection (10-15 cm) in such a short time (around 2 s) the trajectory is smoothed out as a result of high cost of the maneuver.

8.3 Control by Euler angles adjustment

Due to problems, which were described, it was necessary to design another method of control. We decided to control the UAV by adjusting its Euler angles in a time. The principle of this control method is based on prediction of acceleration and yaw angle of the UAV during disturbances. These predictions are then mirrored. Here, we use the fact, that roll and pitch angles are proportional to acceleration around hover operational point of the system. Thus, the acceleration is converted to pith and roll angles and together with predicted yaw angle added to actual Euler angles references.

8.3.1 Pipeline modification

Currently, we were able to send commands to the UAV with desired position of the UAV or command with predicted trajectory. Due to problems, which were described, it was unsuitable to use the current form of the control pipeline. The aim was to avoid the higher control layers. At the same time, we could not discard controllers, which prevents us from regulating identified disturbances. It would be difficult to fly with the UAV then. The promising option was to modify the control pipeline in the lower layers. For this purpose, an adder was added between the attitude controller and the SO(3) controller. Raw data from the SO(3) controller are sent to the first input of this adder. The second input is the euler angles with which we will regulate the influences of the manipulator.

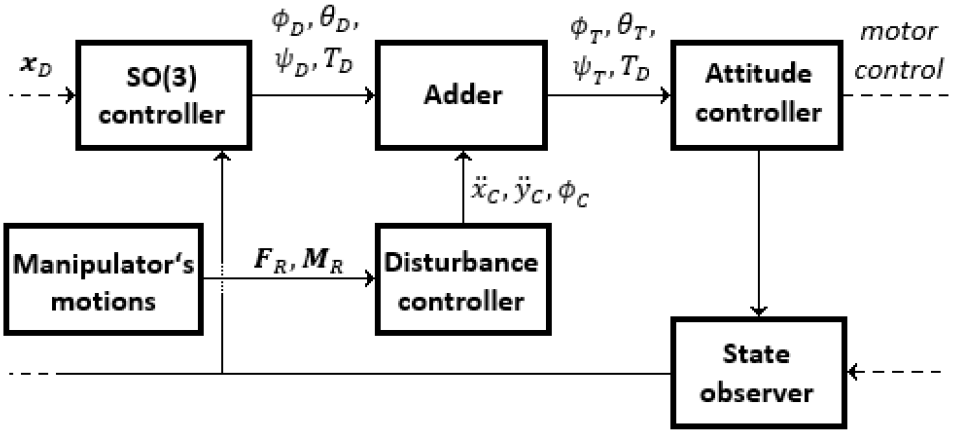


Figure 18: Extended control pipeline diagram. The original pipeline is completed with the adder and disturbance controller.

Due to the fact that the system described above is implemented in ROS, the concrete changes are as follows. A new node was created. This node subscribes messages sent from SO(3) controller to the attitude controller. Another messages that the new node subscribes are messages published by the main node of this thesis. These messages contain desired acceleration in xy-axes and yaw angle by which we want to mitigate identified disturbances. As it was mentioned, dependence between acceleration of the UAV in the xy-plane and its tilt in radians around the UAV operational point is proportional, thus we can write

$$\begin{aligned} \theta_T &= \theta_D + q\ddot{x}_C, \\ \psi_T &= \psi_D + q\ddot{y}_C, \\ \phi_T &= \phi_D + \phi_C, \end{aligned} \tag{55}$$

where θ_T, ψ_T, ϕ_T are the new euler angles, θ_D, ψ_D, ϕ_D are desired euler angles from SO(3) controller, $\ddot{x}_C, \ddot{y}_C, \phi_C$ are accelerations and yaw angle mitigating the predicted disturbances and q expresses approximate ratio between acceleration and euler angle. This coefficient

was determined experimentally as

$$q \approx \frac{1}{6.35}. \tag{56}$$

Such modified euler angles are then published so that the attitude controller can subscribe them. All these described actions work on frequency of 100 Hz. Diagram of the modified pipeline can be seen in figure 18. As result, the Euler angle reference from the SO(3) controller controls the flight of the UAV in space, while the Euler angle correction from disturbance controller mitigate position disturbances caused by the manipulator. The sum of them control both, the UAV and estimated disturbances.

8.3.2 Summary

During simulations we found that the UAV does not reach desired accelerations. We can assume it is due to the higher layers of control, which regulate the position and the Euler angles. The consequence is that the UAV starts to oscillate excessively. This has been solved as follows. Raw accelerations are not sent to the adder directly. Firstly, they are multiplied with sawtooth signal (fig. 19), that gradually weakens desired acceleration. The second positive impact of this product is that the uncertainty on predicted data is suppressed. The gradient of the sawtooth was found experimentally. The second option, to reduce oscillations, is temporary reduction of higher regulator's gains, for the time while the disturbances are present.

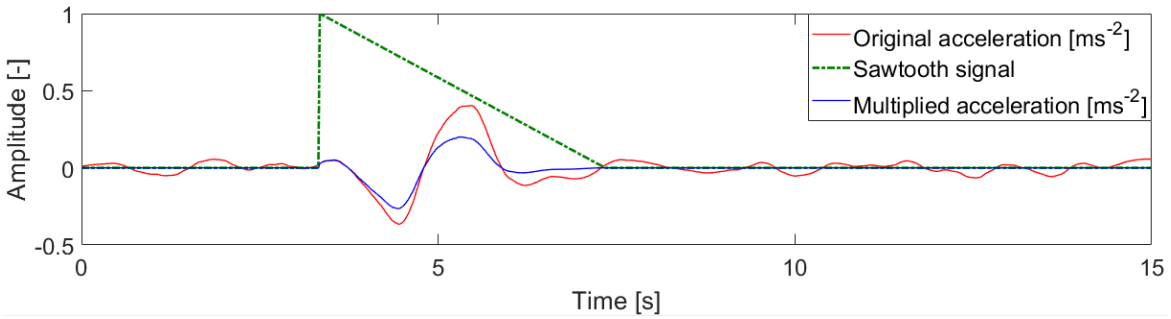
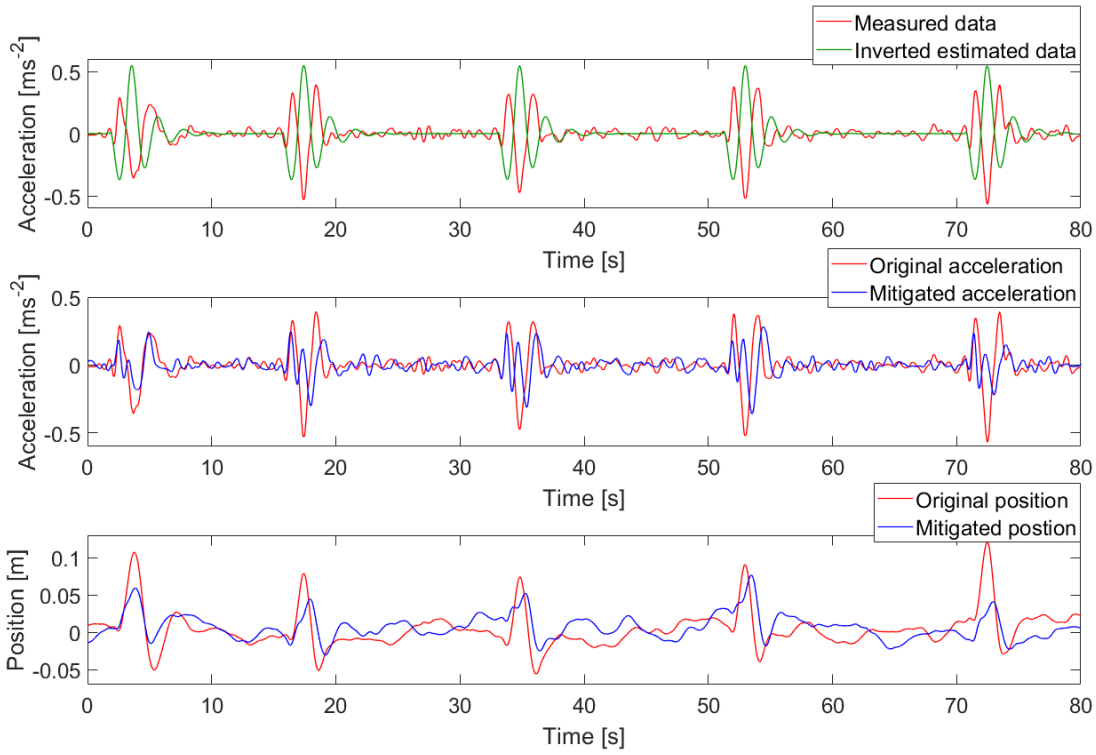
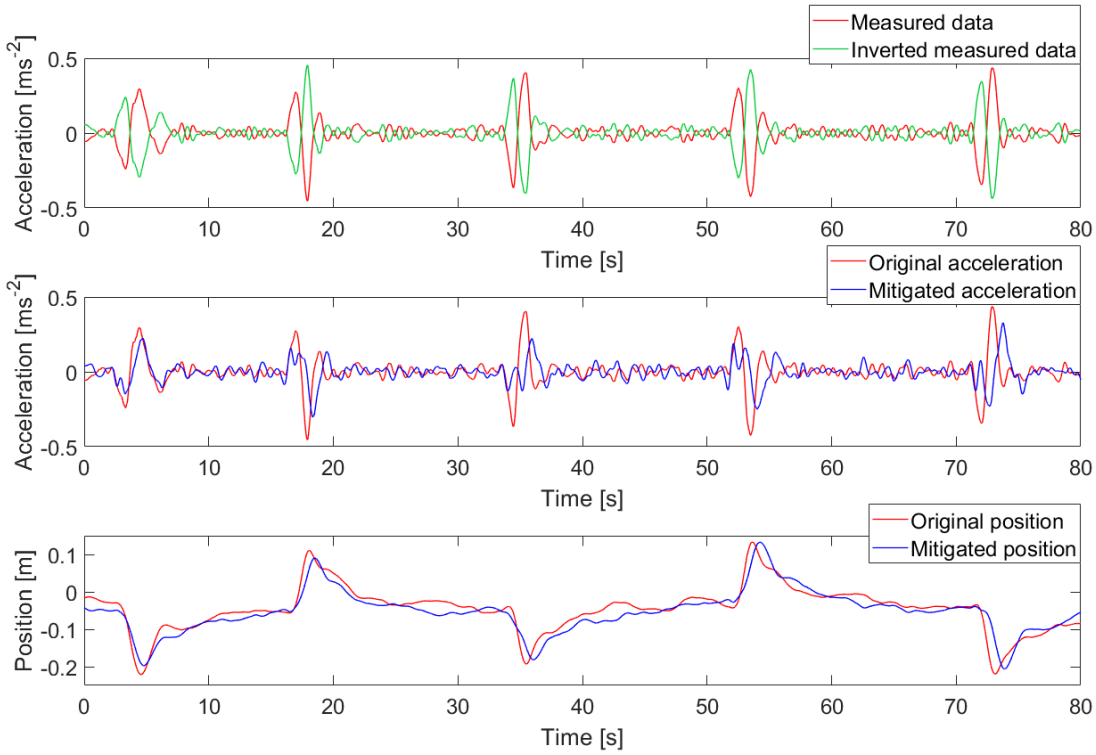


Figure 19: Control signal modification. Multiplication of an acceleration with the sawtooth signal prevents oscillations.

Figure 20a shows results of control in the x-axis. The first subplot shows measured acceleration and mirrored acceleration, that originated from the identified model. This estimated data were then used as control signal for method described above. In the second subplot, there is result of our control. We can notice, that amplitude of acceleration during deflection decreased not negligibly. However, our goal is to reduce influence on the UAV's position. Thus, the most interesting subplot is the last one. We were able to mitigate the position error in the x-axis by 40% on average.



(a) Control of x-disturbance.



(b) Control of y-disturbance.

Figure 20: Control of the UAV's position.

Due to the fact, control by this method is very sensitive on several aspects, as precise timing of control signal, we consider it as positive result. Further, we tried to control the UAV in the y-axis. Due to the situation, when the identified model is not suitable for y-axis, we had to access the backup solution. Mirrored pre-measured acceleration was used as control signal instead of estimated data. This signal was much more noisy then the estimated one. However, this noise should be suppressed by signal multiplication described above. Thus, we expected the same behavior as in the previous case. In the figure 20b, we can see that amplitudes of acceleration were mitigated again. However, there was no improvement in terms of position. Reason of this issue was not revealed, but is somewhere inside the used simulator most likely. We would have to know detailed structure of it for better understanding.

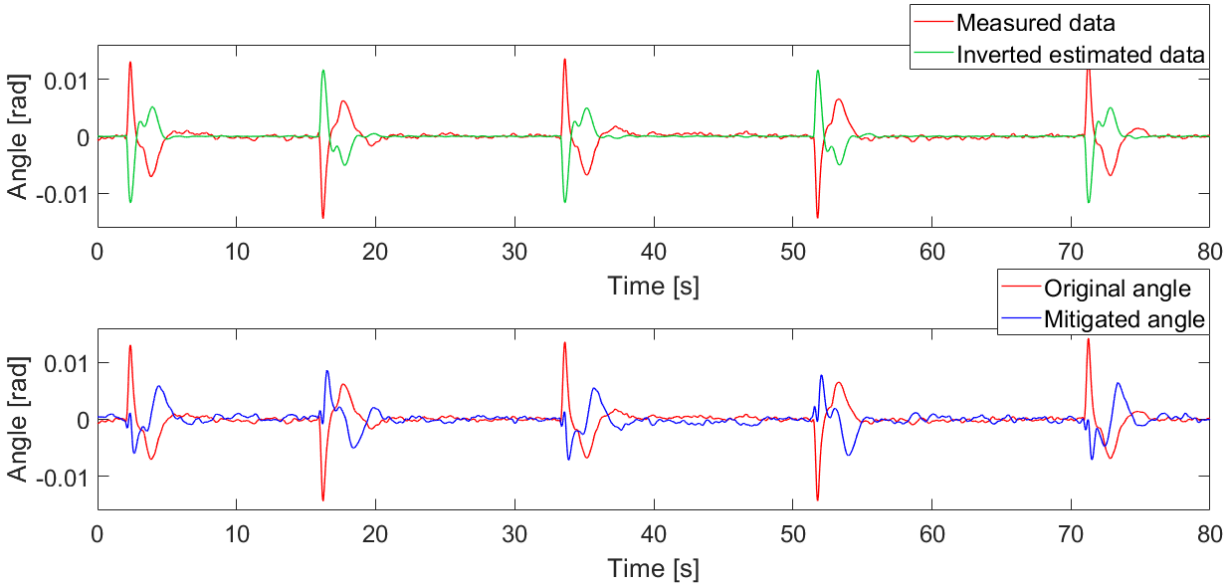


Figure 21: Control of yaw disturbance. We can see an improvement in yaw disturbances.

In the end, control of disturbances in yaw angle was done. We are reaching approximately the same result as in the case of x-axis. More in the figure 21.

8.4 Altitude settling time

As it was described in the previous chapter, the only change in altitude during aerial manipulation consist in rapid increase of settling time. This was fixed by adjustment of parameter, which represents mass of the UAV. The MPC tracker and SO(3) controller operates with this parameter so it is important that the parameter is set correctly. Difference between right and incorrect mass constant is shown in the figure 22.

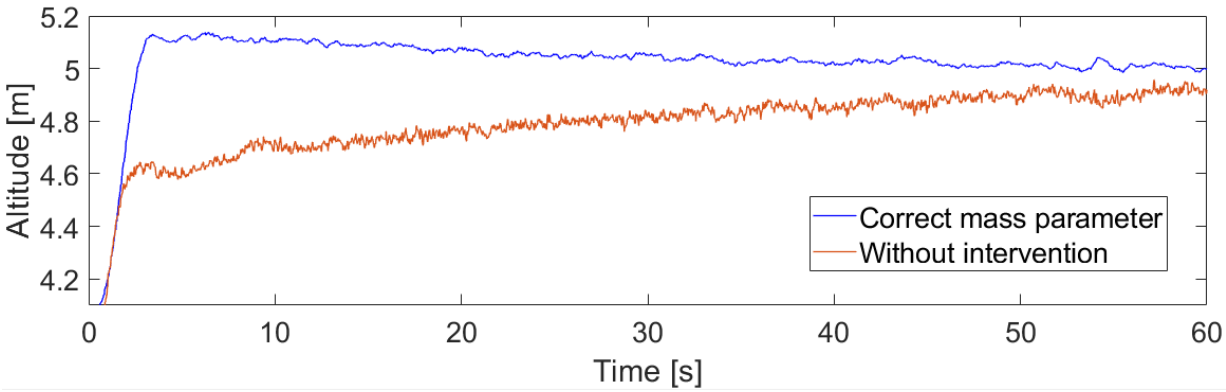
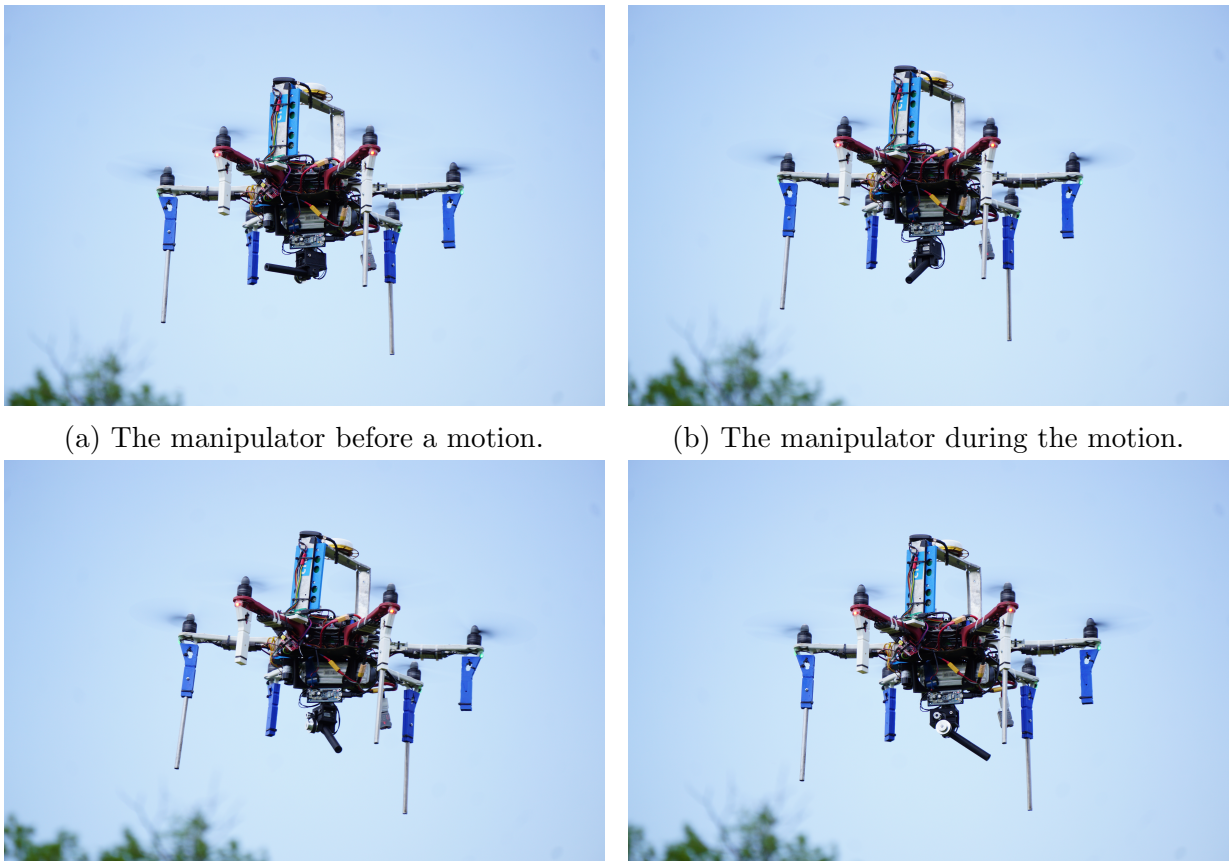


Figure 22: Altitude settling time.

9 Real flight experiment

We tested the hardware prototype in real flight with the UAV. A program in ROS allows controlling the manipulator from the UAV's onboard controller. Thus, we verified that the hardware prototype is integrated within the current software architecture of the UAV. Motions of the manipulator are shown in fig 23.



(a) The manipulator before a motion.

(b) The manipulator during the motion.

(c) It approaches its desired position of the e-e.

(d) The manipulator in the desired position.

Figure 23: Aerial manipulator during the test flight.

10 Conclusion

In this thesis, we have developed a hardware and software solution of 2-DOF manipulator. The hardware prototype has been successfully integrated within the current software architecture of the UAV in ROS. It brings possibility of control the manipulator onboard on the UAV. The manipulator is able to manipulate with objects with high accuracy in case it is equipped with a gripper. However, manipulation with heavy payloads (hundreds of grams) was not satisfactory due to the mechanical construction of used servomotors. For future work it is necessary design a new manipulator with more robust actuators.

The work also created a Gazebo model of the manipulator. The gazebo model of the UAV was equipped with the manipulator model. This system has been tested in simulated experiments. Disturbances caused by the manipulator were found and subsequently identified by mathematical optimization method. This identification was basis of the control of manipulator's disturbances. We designed method of control of these disturbances, which consist of adjustment of Euler angles of the UAV during flight. For this action it was necessary to modify the current control pipeline of the UAV. For identification and control sections, we consider the aerial manipulator as two system - the system of the UAV and the system of the manipulator. Experiments shows, that we are able to mitigate manipulator's disturbances approximately 40%. We consider it as satisfactory result.

According to the assignment, the dynamic model of the UAV was extended. This combined model can be used for more accurate control of disturbances in future work.

References

[1] Tomas Baca, Daniel Hert, Giuseppe Loianno, Martin Saska, and Vijay Kumar. Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles. *Conditionally accepted to 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2018)*, 2018.

[2] Open Source Robotics Foundation. Gazebo simulator. Gazebo official website, 2014. `http://gazebosim.org/`.

[3] Guillermo Heredia, AE Jimenez-Cano, I Sanchez, Domingo Llorente, V Vega, J Braga, JA Acosta, and Aníbal Ollero. Control of a multirotor outdoor aerial manipulator. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 3417–3422. IEEE, 2014.

[4] AE Jimenez-Cano, Jesús Martin, Guillermo Heredia, Aníbal Ollero, and R Cano. Control of an aerial robot with multi-link arm for assembly tasks. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4916–4921. IEEE, 2013.

[5] Suseong Kim, Seungwon Choi, and H Jin Kim. Aerial manipulation using a quadrotor with a two dof robotic arm. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4990–4995. IEEE, 2013.

[6] Taeyoung Lee, Melvin Leoky, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *2010 49th IEEE Conference on Decision and Control (CDC)*, pages 5420–5425. IEEE, 2010.

[7] Giuseppe Loianno, Vojtech Spurny, Justin Thomas, Tomas Baca, Dinesh Thakur, Daniel Hert, Robert Penicka, Tomas Krajnik, Alex Zhou, Adam Cho, et al. Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments. *IEEE Robotics and Automation Letters*, 3(3):1576–1583, 2018.

[8] Nathan Michael, Jonathan Fink, and Vijay Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, 2011.

[9] Mostafa Mohammadi, Antonio Franchi, Davide Barcelli, and Domenico Prattichizzo. Cooperative aerial tele-manipulation with haptic feedback. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5092–5098. IEEE, 2016.

[10] Marc Moonen, Bart De Moor, Lieven Vandenberghe, and Joos Vandewalle. On-and off-line identification of linear state-space models. *International Journal of Control*, 49(1):219–232, 1989.

[11] Jason M. O'Kane. *A Gentle Introduction to ROS*. CreateSpace Independent Publishing Platform, 10 2013. http://www.cse.sc.edu/~jokane/agitr/.

[12] Matko Orsag, Christopher Michael Korpela, Stjepan Bogdan, and Paul Yu Oh. Hybrid adaptive control for aerial manipulation. *Journal of intelligent & robotic systems*, 73(1-4):693–707, 2014.

[13] J Peraire and S Widnall. Lecture l26-3d rigid body dynamics: The inertia tensor. *Dynamics*, pages 1–12, 2009.

[14] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[15] Robot-Shop. Dynamixel xl-320. Robot-Shop eshop, 2015. http://robot-shop.nl/producten/dynamixel-xl-320/.

[16] ROBOTIS. Dynamixel xl-320 manual. Robotis website, 2010. http://support.robotis.com/en/product/actuator/dynamixel_x/xl_series/xl-320.htm.

[17] ROBOTIS. Opencm9.04-c manual. Robotis website, 2010. http://support.robotis.com/en/product/controller/opencm9.04.htm.

[18] Fabio Ruggiero, Miguel Angel Trujillo, R Cano, H Ascorbe, Antidio Viguria, C Peréz, Vincenzo Lippiello, Aníbal Ollero, and Bruno Siciliano. A multilayer control for multirotor uavs equipped with a servo robot arm. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4014–4020. IEEE, 2015.

[19] Ronald W Schafer. What is a savitzky-golay filter?[lecture notes]. *IEEE Signal processing magazine*, 28(4):111–117, 2011.

[20] Marco Tognon, Chiara Gabellieri, Lucia Pallottino, and Antonio Franchi. Aerial comanipulation with cables: The role of internal force for equilibria, stability, and passivity. *IEEE Robotics and Automation Letters*, 3(3):2577–2583, 2018.

[21] William and Wang LLC. Opencm9.04-c. Bannana robotic website, 2013. https://www.bananarobotics.com/shop/Robotis-OpenCM-9.04-B-Microcontroller.

[22] Burak Yüksel, Saber Mahboubi, Cristian Secchi, Heinrich H Bülthoff, and Antonio Franchi. Design, identification and experimental testing of a light-weight flexible-joint arm for aerial physical interaction. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 870–876. IEEE, 2015.

Appendix A CD Content

In Table 3 are listed names of all root directories on CD.

Directory name	**Description**
thesis	Bachelor's thesis in pdf format
thesis_sources	latex source codes
models	STL files for 3D printing
matlab	Matlab scripts for identification
src	executable and supporting files from ROS

Table 3: CD Content

Appendix B List of abbreviations

In Table 4 are listed abbreviations used in this thesis.

Abbreviation	Meaning
ARM	RICS processor architecture
DC	direct current
DOF	degree of freedom
ENU	east-north-up notation
IDE	integrated development environment
MPC	model predictive controller
MRS	multi-robot systems
PID	proportional-integral-derivative controller
ROS	robot operating system
S-G	Savitzky-Golay filter
SNR	signal-to-noise ratio
UAV	unmanned aerial vehicle
URDF	universal robot description format
USB	universal serial bus
Xacro	XML macro
XML	extensible markup language

Table 4: Lists of abbreviations