Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Hybridization of Distance Sensor with Relative Localization System UVDAR

Dominik Fischer

Supervisor: Ing. Viktor Walter
May 2020

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Fischer Dominik**	Personal ID number:	**475380**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Cybernetics and Robotics**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Hybridization of Distance Sensor with Relative Localization System UVDAR

Bachelor’s thesis title in Czech:

Hybridizace dálkového senzoru se systémem relativní lokalizace UVDAR

Guidelines:

The student will familiarize himself with the Robot Operating System (ROS), as well as with the software suite used by the Multi-robot systems group build around ROS.
Some of our platforms use the UVDAR system for mutual relative localization. This system is based on computer vision in the ultraviolet range, and therefore has limited precision in distance estimation. This drawback can be addressed by measuring the distance to a target of known relative bearing with a precise distance sensor.
On the hardware side, the student will select an appropriate distance sensor, microcontroller unit and actuator that will allow a UAV equipped with the UVDAR system to refine the relative pose estimate of a target in terms of distance.
On the software side, the student will develop a driver using ROS, that will allow the distance sensor to reliably aim at targets of known relative bearing retrieved by UVDAR.

Bibliography / sources:

[1] V Walter, N Staub, A Franchi and M Saska. UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs. IEEE Robotics and Automation Letters 4(3):2637-2644, July 2019.
[2] J E Gentle. Matrix transformations and factorizations. Matrix Algebra. Springer, Cham, 2017.
[3] Y Pyo, H Cho, L Jung, D Lim. ROS Robot Programming. Matrix Algebra. ROBOTIS, 2017.

Name and workplace of bachelor’s thesis supervisor:

Ing. Viktor Walter, Multi-robot Systems, FEE

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

Date of bachelor’s thesis assignment: **09.01.2020** Deadline for bachelor thesis submission: **22.05.2020**

Assignment valid until: **30.09.2021**

Ing. Viktor Walter Supervisor’s signature	doc. Ing. Tomáš Svoboda, Ph.D. Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. 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

Acknowledgements

First of all, I would like to express my very sincere thanks towards my supervisor Ing. Viktor Walter for his patience and willingness while guiding me throughout my work regarding this thesis.

My further gratitude belongs to my father, for his advices and all the help he gave me during my studies, and the other members of my family as well, for their constant, not only school–related, support.

Declaration

Abstract

This thesis proposes an extension of an existing system for mutual relative localization of Unmanned Aerial Vehicles (UAVs). The existing system is vision-based and thus suffers from lower precision in distance estimation than in bearing estimation, a challenge that this thesis addresses. The extension consists of an onboard laser distance sensor coupled with a positioning mechanism. The sensor is then used for retrieving mutual distance between UAVs. This thesis is mainly focused on the design and assembly of the positioning mechanism for the sensor and a subsequent development of a driver for the positioning mechanism. Said driver will also allow processing of the data received from the sensor. Finally, the developed system is tested in the Gazebo simulator.

Keywords: relative mutual localization, UAV, distance measurement, ROS, simulation, Gazebo

Supervisor: Ing. Viktor Walter
Czech Technical University
Faculty of Electrical Engineering
Department of Cybernetics
Karlovo náměstí 13
121 35 Prague 2
Czech Republic

Abstrakt

V této práci je navrženo rozšíření existujícího systému pro vzájemnou relativní lokalizaci bezpilotních helikoptér (UAV). Stávající systém je založen na počítačovém vidění, a proto je méně přesný při určování vzdálenosti než směru. Předkládaná práce se proto věnuje zvýšení přesnosti určování vzdálenosti pomocí laserového senzoru. Ten je umístěn na pohyblivém závěsu přímo na palubě helikoptéry a umožňuje přesné měření vzdálenosti mezi bezpilotními drony. Náplní práce je návrh a sestavení závěsného mechanismu pro laserový senzor a vývoj obslužného programu, který mechanismus řídí a zpracovává získaná data. Celý systém je testován v Gazebo simulátoru.

Klíčová slova: relativní vzájemná lokalizace, UAV, měření vzdálenosti, ROS, simulace, Gazebo

Překlad názvu: Hybridizace dálkového senzoru se systémem relativní lokalizace UVDAR

Contents

Figures

Tables

Chapter 1

Introduction

The term Unmanned Aerial Vehicle (UAV) denotes an aircraft that does not have a human pilot onboard and is operated either remotely or autonomously by onboard computers. Their size varies from large military UAVs such as the Heron TP designed for the Israeli Air Force, with wingspan of 26 m and take–off weight of 4 650 kg [1], to so called Micromechanical Flying Insect (MFI) with less than 2.5 cm wingspan [2]. In this thesis the term UAV refers to a quadrotor DJI F450, see Fig. 1.1. UAVs, in general, started being systematically developed by the U.S. Army in World War I. [3]. However, nowadays their field of application extends widely beyond the military deployment. Despite being considered unreliable and inaccurate at first [4], thanks to technological advancement, UAVs are being used more and more often for a large number of different tasks and are, therefore, becoming a very promising field for study and research.

Despite the range of applications of UAVs being relatively wide, certain limitations to what one drone can accomplish at its own do exist. The payload can be too heavy to carry or with too much inertia to stabilize. The task may be too complex for a single unit to accomplish, or this could take an unreasonable amount of time. Possible solutions to these problems are being assiduously researched by the robotics community and one of the proposed approaches is to employ multiple UAVs at once or even UAV swarms in order to complete the given task. The success of said cooperation is conditional on the ability of the UAVs to locate each other and thus being able to adjust their behaviour and actions accordingly.

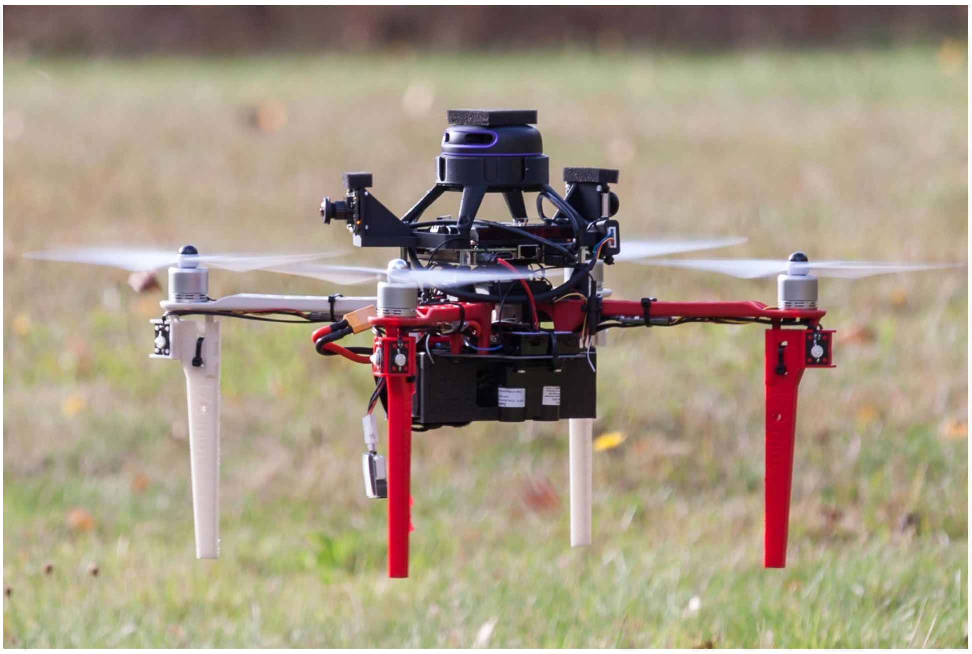


Figure 1.1: The DJI F450 quadrotor [5].

In light of the above, a reliable system for mutual relative localization of UAVs is vital in every application, where some form of cooperation between multiple UAVs is expected. The Multi-robot-Systems (MRS)[1] Group from Czech Technical University (CTU) in Prague are addressing the problem of mutual relative localization of multiple UAVs by using a system called UVDAR (Ultraviolet Direction and Ranging) [6]. The UVDAR system is based on computer vision in the ultraviolet range and consists of ultraviolet blinking LED markers and two UV cameras per UAV. The importance of accurate distance estimate in systems for mutual localization of not only flying machines is self-evident. A more formal proof of this statement can be found in [7], where two systems for anonymous mutual localization of UAVs were examined and compared. The first system used bearing-only measurements and the second one combined the bearing and distance measurement. The localization system that used both bearing and distance measurements was shown to perform significantly better.

The main drawback of the UVDAR system at the moment is that a reliable and precise distance between two or more UAVs is difficult or in some cases even impossible to retrieve due to a finite resolution of the UV cameras used in the UVDAR system. This problem can be solved by integrating an external distance sensor that will reliably measure distance to a target of known relative bearing retrieved by UVDAR. In this thesis, such onboard sensor is integrated into the UVDAR system, together with a positioning

[1]https://mrs.felk.cvut.cz

mechanism that will allow the sensor to be aimed at targets of interest and retrieve their distance relative to the UAV carrying the sensor. Despite the fact, that in the majority of multi–UAV applications, more than just two UAVs are employed, in this thesis the situation is restricted only to a pair of UAVs. A functional system, that can reliably localize one target UAV can be seen as a solid stepping stone in order to widen and adjust the system so that it can track multiple targets at once. Along with the hardware side of the proposed extension a driver is developed that will move with the designed positioning system and hence aim the sensor. The driver is implemented using ROS (Robot Operating System), which is a widely used meta-operating system that offers a development environment for robot programming, where multiple hardware and software components can be connected [8].

The outline of this thesis is as follows. In the State of the Art chapter, the recent findings and research in the field of UAV localization are overviewed, followed by a chapter where the theoretical analysis of the proposed extension is provided. Next, the hardware side of the proposed actuator and sensor is described. A description of the developed software driver is provided in the Software Driver Development chapter and lastly, the assembled prototype with distance sensor driven by the developed driver is tested in simulation.

Throughout this thesis, some of the mathematical operations require the use of matrices and vectors. Overview of the used notation and symbols is in Table 1.1.

Table 1.1: Used matrix and vector notation

Symbol	**Meaning**
$\mathbf{T}_j^i$	Transformation from coordinate frame i to coordinate frame j
$\mathbf{v}_j$	Vector expressed in coordinate frame j
$\mathbf{\Sigma}$	Covariance matrix

Chapter 2

State of the Art

Since the problem of mutual localization is vital for any autonomous application of UAVs, it has been and is continuing to be the focus for many research teams all over the world. The most obvious approach is to rely on absolute measurement as a source of data for position retrieval. Numerous indoor as well as outdoor examples can be found in literature. The majority of the laboratory experiments use motion capture system (Optitrack[2], VICON[3], etc.) as a source of information. For instance, in [9] the VICON motion capture system is used for localization of swarms of micro UAVs and commands for each UAV are centrally calculated on a desktop station based on the received data. Outdoor cooperative flights tend to rely on Global Navigation Satellite System (GNSS) as in [10] where each UAV is equipped with a GPS receiver and shares its position with other flock members. The main disadvantage of the aforementioned approaches is the need to preinstall the needed infrastructure, and, therefore, the inability to be used in inaccessible areas.

A decentralized approach addresses this issue by replacing absolute position measurements with onboard sensors. Solutions presented in literature vary mostly in used sensors and measured parameters. An example of such approach, combining a visual sensor with inertial measurements from the robot inertial measurement unit (IMU), can be found in [11]. The disadvantage of the presented solution is that the UAVs rely on communication that can be subject to network congestion and interference or outdoor conditions in general and that the localization algorithm is computationally expensive. A purely vision-based absolute localization system that does not rely on

[2]https://optitrack.com/
[3]https://www.vicon.com/

inter-UAV communication is proposed in [12]. The UAV compares captured pictures of the ground beneath it and compares them to a database of stored georeferenced frames. To increase robustness of this method, which is otherwise highly susceptible to errors caused by season changes or light conditions in general, the Mutual Information method [13] is used. In order to suppress the otherwise inevitable errors in vision-based absolute position estimation that occur due to the constant environmental and weather changes, the measurement of relative position can be employed. A simple method of mutual relative localization of two UAVs is described in [14] where the only information used for determining the relative positions of both UAVs are the measured distances from one another and a desired trajectory of each UAV. Even though this approach is sufficient for simple trajectories, it can not be used for more complex applications since the proposed algorithm is described only for the case where both UAVs are traveling at equal altitudes.

The approach used in this thesis is based on vision in the ultraviolet spectrum and mitigates the majority of the aforementioned drawbacks. For a detailed description of the Ultraviolet Direction and Ranging system see [6, 15, 16]. The main advantages of using the UVDAR system for mutual relative localization are twofold. First, due to the system working in the ultraviolet spectrum it is very robust and reliable regardless the light and environment conditions. Secondly, it uses only light onboard cameras and is not as demanding in terms of computational power as the system proposed in [11], making it an easily embeddable system that does not need any external infrastructure. Furthermore, the UVDAR system was shown to perform well in complex outdoor conditions and experiments, which can be found in [17].

Chapter 3

Theoretical Background

As mentioned in the introduction, the UVDAR system is not precise in terms of distance estimation. The proposed onboard extension of the UVDAR system that addresses and aims to solve this problem is described in this chapter. First, the UVDAR system itself is briefly introduced with its main functionalities and working principles. This is followed by a basic description of the proposed UVDAR extension. After that the coordinate systems connected to different parts of the developed mechanism are described, together with transformations between them. Then the coordinates describing the state of the system, namely the yaw and pitch angles are derived. Last but not least, the way of using the measured distance in order to make the estimated positions more precise is presented.

In this chapter the matrix and vector notation introduced in Table 1.1 is used.

3.1 Ultraviolet Direction and Ranging system

The Ultraviolet Direction and Ranging (UVDAR) system is an onboard system for mutual relative localization of UAVs. It was developed by the Multi-robot-Systems Group at CTU in Prague. The system consists of active ultraviolet LED markers on the UAVs and cameras that can detect ultraviolet light. Since the system consists only of these easily embeddable parts, one of its main advantages is evident. The system does not need any external equipment or special preparation in the operating area. Furthermore, due to UVDAR working in the UV spectrum, the system is robust with respect to environmental conditions such as bright sunlight or complex background. One of the main working principles of UVDAR is based on the observation that the presence of ultraviolet light in sunlight is significantly lower than the colors in visible spectrum. Not only that, but additionally not many artificial sources of light emit ultraviolet light as well. This means that when the camera captures an UV source, it is most likely a blinking marker attached to a target UAV. The only other frequently present source of UV light is the sun and the disturbance that it presents can be easily suppressed with the knowledge of the size and position of the spots in camera image. These properties of UVDAR make it a system suitable not only for indoor and laboratory applications with controlled lighting, which is usually of high importance for any computer vision-based experiments, but it can be successfully used outdoors regardless of the weather and lightning conditions [6].

The blinking LED markers are used not only for relative position estimation but their presence is utilized further. As thoroughly described in [16] the blinking pattern can be also used to retrieve relative heading of the target UAV and as a means of identification. The blinking LEDs have a known layout that ensures that from every viewpoint at least two markers are visible. Half of the LEDs are mounted on the port side of the UAV and the other half on the starboard side. LEDs on each side are blinking with a different frequency making it possible to unequivocally retrieve the relative yaw of the target UAV with respect to the observer. The relative bearing can be assessed from the combination of the number of visible markers and their blinking frequencies.

The above working principle and properties make UVDAR a reliable, robust and undemanding, in terms of required equipment, system for relative mutual localization of UAVs that can be used outdoors and even in unknown areas and areas that are difficult to access. The UAVs using this system are able to fly in formations, such as leader-follower or classical swarms, for more information on flying formations see [16] and [18].

3.2 Basic Structure of the Proposed Extension

One of the main tasks of any localization system is to be able to determine how far the localized object is. It is desirable for the UAV to be able to measure distance to other objects without the need to adjust or change its planned trajectory. It is therefore necessary for the rangefinder to have a mechanism that allows it to be aimed independently of the pose of the UAV.

The proposed onboard sensor can be divided into two main parts. One is the distance sensor itself and the second one is the necessary positioning mechanism. The positioning mechanism has two degrees of freedom (2-DOF) and can be unequivocally described using two angles, which will be called by terms borrowed from aircraft principal axes as yaw and pitch $[\phi \;\; \psi]^T$. The 2-DOF allow the sensor to be rotated horizontally as well as vertically, ensuring that all targets retrieved by UVDAR can be aimed at. The used distance sensor is an optical laser-based distance measurement sensor of which a more in depth description is provided in Subsection 4.1.3.

In order to control the above hardware, a software driver was developed and is discussed in Chapter 5.

3.3 Coordinate Frames

Since the positioning mechanism consists of multiple parts, multiple coordinate frames are needed for the description of relative position of targets with respect to different parts of the manipulator. A basic diagram of the complete system, together with individual coordinate frames and their positions is shown in Fig. 3.1. The coordinate systems are the following:

- The central coordinate system in which the input coordinates are expressed is connected to the body of the UAV. It has origin $\mathbf{O}_u$ and is denoted by subscript u.
- The UV camera has a coordinate system denoted by subscript c. In Fig 3.1 a simplified model is shown with only one camera. In reality, two UV cameras are mounted onto the UAV in order to widen the field of view. Each of the cameras has its own coordinate frame and origin. The origin of the camera frame in the simplified diagram is denoted by $\mathbf{O}_c$.
- The origin $\mathbf{O}_h$ of the coordinate frame connected to the servo rotating in horizontal plane lies in the center of its rotor and denoted by index h.

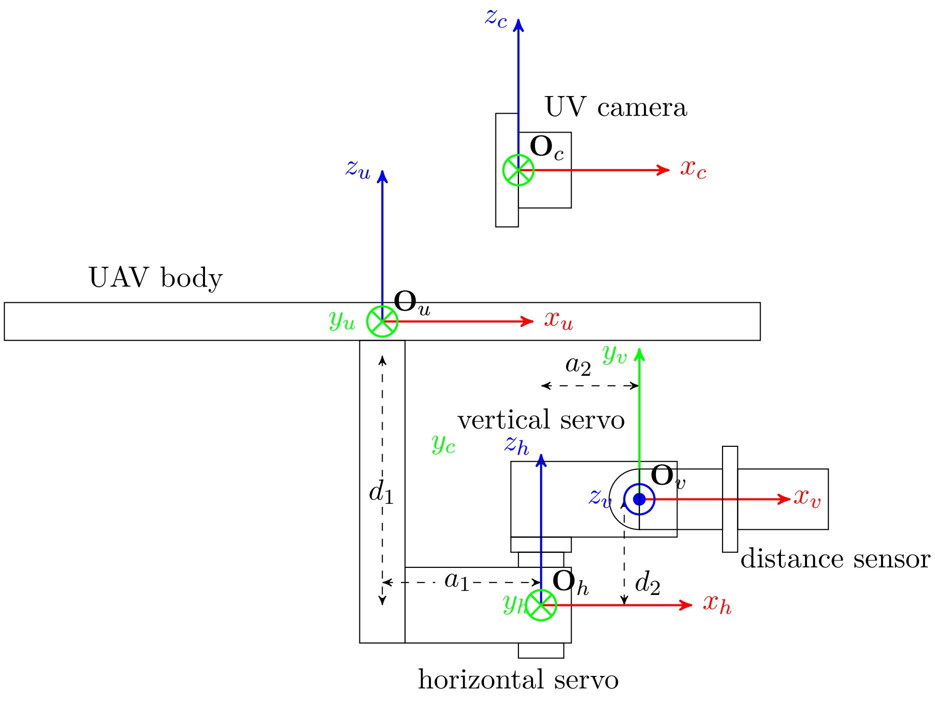


Figure 3.1: A side view of the individual coordinate systems connected to the UAV together with their positions.

- The vertical servo coordinate frame is accompanied by subscript v and its origin $\mathbf{O}_v$ is located in the center of the rotor rotating in vertical plane. Since both the coordinate frame as well as the distance sensor are fixed to the rotor, the vertical servo coordinates can be used to describe the sensor as well.

The camera frame from Fig. 3.1 is superfluous to the described application and is mentioned only for completeness. The target 3D position computed by the UVDAR is considered as reliable for the purpose of this work. For more information regarding obtaining the position of the target from the visible markers, including its filtering and stabilization, the reader is reffered to [15].

3.3.1 Coordinate Transformations

In order to be able to switch between coordinate systems when expressing coordinates of a point in space, the relations between individual coordinate frames and their positions need to be described.

Let $\mathbf{v}_u = [x_u \;\; y_u \;\; z_u \;\; 1]^T \in \mathbb{R}^4$ be the input vector of homogeneous coordinates in the UAV body frame. In the interest of future computations it is desirable to express the coordinates in different coordinate systems as well. The derivation of transformational relations between individual frames follows.

Every transformation in space can be broken up into translation and rotation around its coordinate axes, which are representable by matrices [19]. In this instance the parameters and matrices of each transform were found using the Denavit-Hartenberg (DH) notation which restricts movements to only four possible actions described by four DH parameters α, a, θ and d whose meanings are explained in Table 3.1. Each transformation between two distinct frames can thus be described using these parameters.

Table 3.1: Table of DH parameters with meaning.

Parameter	**Meaning**
θ	rotation around the z axis
d	translation along the z axis
α	rotation around the x axis
a	translation along the x axis

Firstly, in order to get from the UAV body frame to the coordinate frame connected to the horizontal servo, the origin $\mathbf{O}_u$ needs to be shifted by $-d_1$ along the z_u axis and then by a_1 along the x_u axis. The 4×4 transformation matrix $\mathbf{T}_u^h$ that performs said shifts is in (3.1).

$$\mathbf{T}_u^h = \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

In order to get from the horizontal servo frame to the vertical servo frame, a slightly more complex transformation needs to be performed. First step is to rotate the coordinate system by ϕ around the z_h axis then shift by d_2 along the same axis and finally shift by a_2 in direction of the new rotated axis x'_h. For the meaning and definition of angle ϕ see Section 3.4. Because according to the DH-notation, the joint should rotate around its z axis, it is necessary to perform one last rotation around the x_v axis by $\frac{\pi}{2}$. The form of the translation matrix was already introduced in (3.1) and the general form of a rotation matrices around the x axis is in (3.2) and around the z axis is in (3.4).

$$\mathbf{R}_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) & 0 \\ 0 & \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$\tag{3.3}$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

To sum up, the matrix $\mathbf{T}_h^v$ that performs the transformation between the horizontal and vertical servo frames is derived in (3.5) as a product of two rotational matrices and one translational.

$$\mathbf{T}_h^v = \mathbf{R}_z(\phi) \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{R}_x\left(\frac{\pi}{2}\right). \tag{3.5}$$

The transformation

$$\mathbf{T}_u^v = \mathbf{T}_u^h \mathbf{T}_h^v, \tag{3.6}$$

takes the coordinates in the vertical servo frame and returns its coordinates in the UAV body frame and its DH parameters are in Tab. 3.2.

Table 3.2: Values of DH parameters

Final frame	θ	d	α	a
Horizontal servo	0	$-d_1$	0	a_1
Vertical servo	ϕ	d_2	$\frac{\pi}{2}$	a_2

However, what is desired, is the inverse transform, because the input is in the UAV body frame. Therefore, the position of the target in the vertical servo frame $\mathbf{v}_v$ is obtained using relation (3.7).

$$\mathbf{v}_v = \mathbf{T}_v^u \mathbf{v}_u = \left(\mathbf{T}_u^v\right)^{-1} \mathbf{v}_u, \tag{3.7}$$

where $\mathbf{v}_v$ and $\mathbf{v}_u$ are homogeneous coordinates in the vertical servo and UAV body frames. $\mathbf{T}_u^v$ and $\mathbf{T}_v^u$ are transformation matrices.

3.4 Obtaining Servo Angles

The success or failure of the developed distance measuring system depends on how accurately the individual servo angles will be calculated and thus how precisely will the sensor be aimed.

As mentioned in Section 3.2, the positioning mechanism has two degrees of freedom and each configuration can be described using the coordinates ϕ and ψ. The values of ϕ and ψ for every input set of cartesian coordinates $\mathbf{v}_u = [x_u \;\; y_u \;\; z_u]^T$ can be calculated by solving an inverse kinematic task for the manipulator, whose structure is in Fig. 3.2. Note that here the values x_u, y_u and z_u denote the distinct coordinates and not the axes themselves.

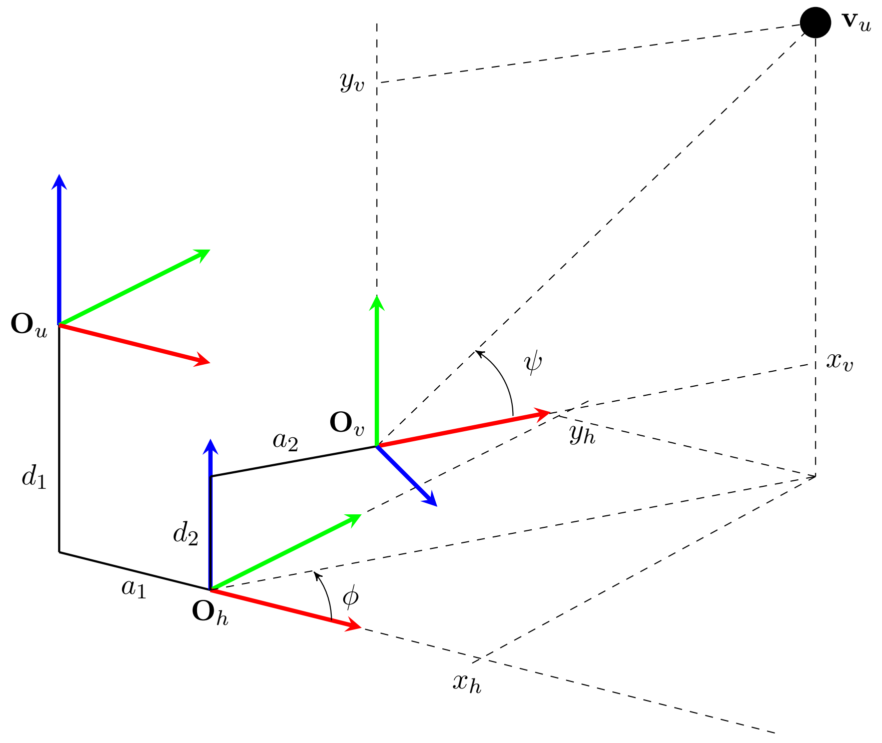


Figure 3.2: Graphical expression of individual actuator coordinates ϕ and ψ. Axis follow the $xyz :=$ RGB convention.

The angle ϕ can be calculated by substituting coordinates $\mathbf{v}_h$ into the relation (3.8).

$$\phi = \operatorname{atan2}(y_h, x_h), \tag{3.8}$$

where x_h and y_h are the target coordinates expressed in the horizontal servo frame.

The pitch angle ψ is then calculated using the input vector expressed in the vertical rotor frame. The pitch angle ψ is then

$$\psi = \operatorname{atan2}(y_v, x_v), \tag{3.9}$$

where x_v and y_v are the target coordinates in the vertical rotor frame.

3.5 Distance with Covariance

The main purpose of the developed rangefinder is to enhance the precision of target position estimation done by the UVDAR. The way the measured distance is used to achieve that is described in this section.

The UVDAR uses a Kalman filter for estimation of the states that the target is currently in. However, the Kalman filter can be also used for sensor data fusion [20], which can be exploited in this instance. For a detailed description of a Kalman filter and its capabilities, see [20] or [21]. The principle of sensor fusion is to combine multiple sensor measurements, here namely the UVDAR estimation and the measured distance, in order to make the estimation more precise.

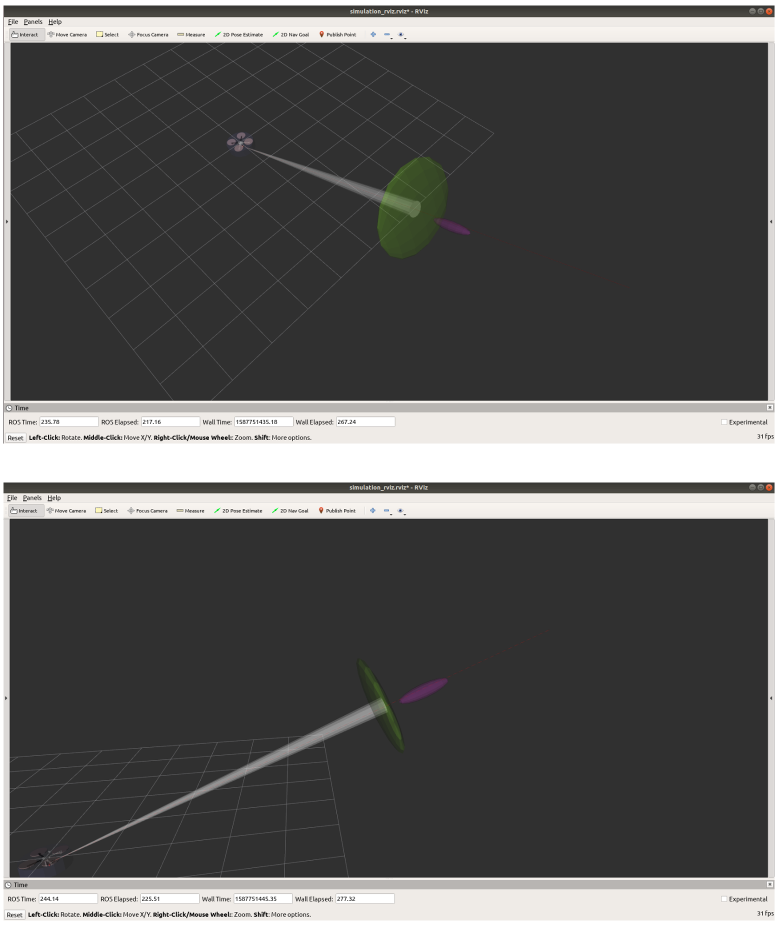
The next task is to extract a position estimation of the target from the measured distance. When the distance sensor makes an acquisition, it means that the positioning mechanism is in such a configuration that the x_v axis goes directly through the target. This means that the position of the target in the vertical servo frame is $\mathbf{v}_v = [l \;\; 0 \;\; 0]^T$, where l is the measured distance.

In order to successfully fuse the measurements with Kalman filter the uncertainty of the said estimation needs to be provided. Let the measured distance be a random variable with Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2. With the rangefinder having an accuracy of ± 2.5 cm the distribution is $\mathcal{N}(l, 0.025)$ [m]. Covariance measures the joint variability of two random variables and while having multiple random variables a convenient way of representing these relations is with help of a covariance matrix $\boldsymbol{\Sigma}$. The 3×3 covariance matrix corresponding to the distance measurement is

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma^2 & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{bmatrix} = \begin{bmatrix} 0.025 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.10}$$

where r represents an arbitrary variance in heading that allows for seamless inclusion in the Kalman filter. The value of r is set to 1.

The elements $\mathbf{\Sigma}_{ij}$ stand for how much the *ith* a *jth* elements correlate together. The fact that here $\mathbf{\Sigma}$ is diagonal means that there is no correlation between the uncertainty in distinct coordinates. Covariance represented by matrix (3.10) can be visualized as a flattened disk, see Fig. 3.3, whose width–a graphical representation of distance uncertainty is much smaller than its radius. As can be seen in Fig. 3.3, the proposed extension has much lower variance in terms of how far the estimated position is.



ROS Time: 235.78 ROS Elapsed: 217.16 Wall Time: 1587751435.18 Wall Elapsed: 267.24

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 31 fps

ROS Time: 244.14 ROS Elapsed: 225.51 Wall Time: 1587751445.35 Wall Elapsed: 277.32

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click/Mouse Wheel:**: Zoom. **Shift**: More options. 31 fps

Figure 3.3: Graphical representation of covariances of position estimated by the UVDAR (pink) and by the proposed extension (green).

Pairing the vector $\mathbf{v}_v$ together with $\boldsymbol{\Sigma}$ gives the position estimation that is more precise in terms of distance than the estimation made by UVDAR alone. This is then sent to the Kalman filter.

It can occur that the rangefinder misses the target and the measured distance is incorrect. Then it is desired to discard such measurement and rely solely on the UVDAR estimation. Such case is detected by calculating the Mahalonobis distance between the estimated position and the state predicted by the Kalman filter. Mahalonobis distance is defined by equation (3.11).

$$d_M = \sqrt{(\mathbf{x}-\mathbf{y})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\mathbf{y})}. \tag{3.11}$$

The Mahalonobis distance can be used to measure dissimilarity between two random vectors $\mathbf{x}$ and $\mathbf{y}$ of the same distribution. Should such an event occur that the estimated position provided by the proposed distance measuring system is too distant in terms of Mahalonobis distance, it is discarded, because the laser beam most likely did not hit the target at all. Otherwise an update of the Kalman filter state is done using the estimation.

Chapter 4

Hardware Assembly

In this chapter, the hardware side of the designed UVDAR extension is described. Firstly, a list of used components is provided, followed by a detailed description of each part. In the last section, the proposed distance measuring mechanism is assembled and described, together with its properties.

Some of the abbreviations used throughout this chapter are left unexplained here. Their meaning can be found in Table A.1.

4.1 Used Hardware Components

As mentioned in Chapter 3, the proposed extension of the UVDAR system consists of a distance laser sensor and a positioning mechanism. The key hardware parts that make the onboard sensor are the following:

- OpenCM9.04-C microcontroller board
- two Dynamixel AX-12A servo actuators
- Garmin LIDAR-Lite v3 laser distance measurement sensor

4.1.1 OpenCM9.04-C Microcontroller

OpenCM9.04-C is a microcontroller board from the manufacturer ROBOTIS[4] based on 32bit ARM Cortex-M3 mounted namely with a STM32F103CB CPU with 128K bytes of Flash, 20K bytes of SRAM and a full set of GPIOs. It also features UART, JTAG/SWD Serial Wire Debug, SPI and I2C interface. The connection to a computer can be estabilished using a micro B USB (Universal Serial Bus), and the board can be programmed using an OpenCM IDE[5], which is a development software and download tool created especially for the OpenCM boards. However, because the OpenCM IDE is not being maintained anymore, the use of Arduino IDE[6] is encouraged. Numerous libraries and example programs exist and the board can be used to control different types of hardware and different applications, making the work with the OpenCM9.04-C similar to Arduino[7] boards. The OpenCM9.04-C board has four TTL bus connectors that can be used for direct connection of the used Dynamixel servos. The controller operates at 3.3 V but also offers the option of connecting an external power supply. It has an operating voltage between 5 V–16 V, making it possible to power said servos through the board with connected voltage supply [22]. The direct compatibility with the Dynamixel servos and the variety of supported development platforms with examples and libraries were the main reasons for using this board as the controller for the positioning mechanism.

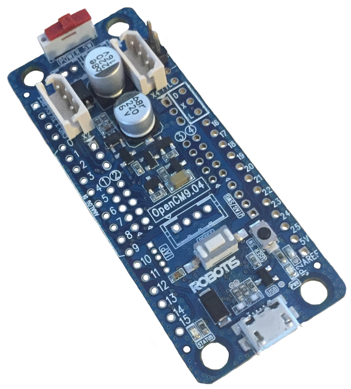


Figure 4.1: OpenCM9.04-C board.

[4]http://www.robotis.us/
[5]http://emanual.robotis.com/docs/en/software/opencm_ide/getting _started/
[6]https://www.arduino.cc/en/main/software
[7]https://www.arduino.cc/

4.1.2 Dynamixel AX-12A Servo Actuator

The aiming mechanism uses two Dynamixel AX-12A[8] servos also manufactured by ROBOTIS. The use of these servos is justified by the high resolution of 0.29 degrees in position and its low weight of 54.6 g. The requirement for low weight is crucial in designing all parts of the UAV, since only limited mass can be attached without negatively impacting its flying capabilities. The AX-12A is currently one of the most advanced servos available. It has the ability to track its speed, temperature, shaft position, voltage or load. It uses TTL Half Duplex Asynchronous Serial protocol for communication so it is possible to send commands to the servo as well as read feedback values. Each servo can be distinguished by unique ID making it possible to control multiple servos with a single microcontroller and connect them together. The AX-12A servo operates on 9 V–12 V with a current flow of 1.5 A at maximum load. As mentioned in Section 4.1.1 the servos can be controlled using special libraries and commands or directly by writing values into specific address in the RAM or EEPROM area. The servo can rotate both directions and operate in two modes–Joint Mode and Wheel Mode [23]. In the Wheel Mode the rotation of the servo is not constrained by any limits and thus it can turn endlessly. Joint Mode means that two limits of rotation are set, each in one direction, and thus the movement of the servo is restricted. In the application described in this thesis, the servos were both used in Joint Mode.

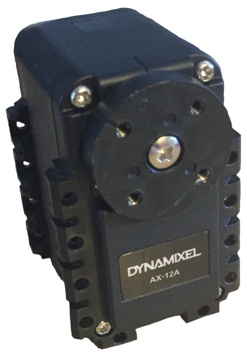


Figure 4.2: Front view of the Dynamixel AX-12A.

[8]http://emanual.robotis.com/ docs/en/dxl/ax/ax-12a/

4.1.3 Garmin LIDAR-Lite v3

The core of the proposed relative distance measuring mechanism consists of a distance sensor. For the researched application the optical distance measurement sensor LIDAR-Lite v3 manufactured by Garmin[9] was selected. This sensor, shown in Fig. 4.3, is based on the time-of-flight principle using a near-infrared laser. For more detailed description of the operational theory behind the sensor measurements see the Theory of Operation subsection of this subsection. The LIDAR-Lite v3 was chosen primarily due to its compact size and low weight of 22 g. Furthermore its performance is very high. The most important performance attributes are range up to 40 m with resolution of ±1 cm and accuracy of ±2.5 cm when measuring distances that are greater than 5 m. With distances lower than 5 m the accuracy decreases down to ±10 cm. However due to the expected deployment field of the researched application being outdoors, where the relative distances tend to be higher, the accuracy and performance of the selected sensor is considered to be more than sufficient. The sensor operates on 5 V and requires an external power source. A microcontroller operating the sensor and taking care of the measurements is also needed. An Arduino Mega 2560 was used in the development and ground testing stage but after mounting the complete measuring mechanism aboard the UAV, the LIDAR-Lite v3 will be connected directly to the onboard computer. There are two basic communication configurations for this sensor, namely a two-wire I2C-compatible serial interface and PWM. The sensor also allows the user to configure it and customize its performance in terms of accuracy, operating range and measurement time depending on the application. The LIDAR-Lite v3 features an edge-emitting, 905 nm - 1.3 watt single strip laser transmitter exhibiting a beam divergence of 8 mrad [24]. Beam divergence is an angular measure of the increased beam diameter/radius with distance from the source. The very small value of 8 mrad implies that a precise aiming and positioning mechanism has to be coupled with the sensor.

Theory of Operation

Detailed operational and measurement principles of the LIDAR-Lite v3 are described in this subsection.

The time-of-flight principle mentioned in 4.1.3 means that the sensor uses measured time delay between the transmission and reception of the laser signal and the known speed of the emitted light to determine the distance to the measured object.

[9]https://www.garmin.com/en-US/

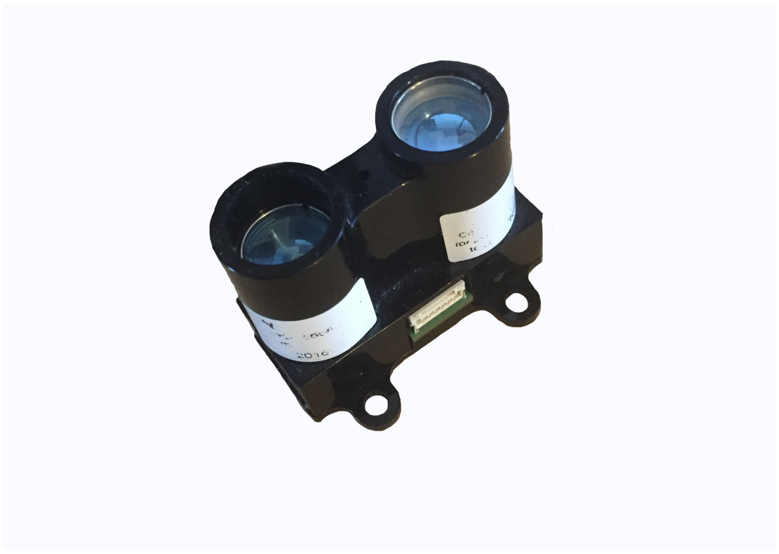
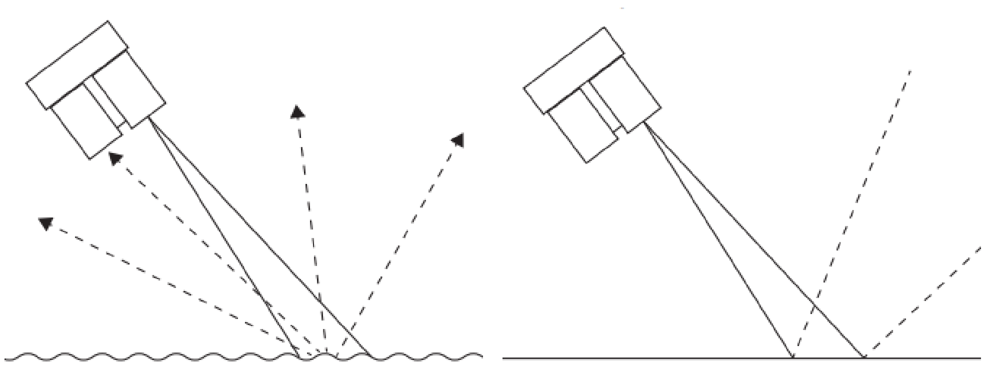


Figure 4.3: Garmin LIDAR-Lite v3.

Before taking a measurement, the device first performs a correction routine, correcting for different ambient light conditions. After that a calibration for "zero" distance needs to be done. The sensor sends a reference signal directly to the receiver and stores the measured time delay as a value for "zero" distance. This value is repeatedly recalculated after several measurements.

The measurement itself consists of multiple transmissions of the laser signal. If a matching signal to the transmitted laser is received it is stored to memory as a correlation record. When an object at certain distance reflects the signal back to the device, the recorded matching acquisitions are summed together and cause a peak to emerge at the corresponding distance location at the correlation record. After such peak is detected, the device calculates the distance and reports it in cm. If the returned signal is not strong enough and the device reaches the predetermined amount of acquisitions without detecting a peak, it stops the measurements and returns a default value of 1 cm. Before the next measurement the device erases all the received and stored data and starts again [24].

A non negligible aspect that affects the ability to reliably measure distance is the reflectivity of the surface of the target. Two types of surface with different reflective characteristics and their effect on measurements with the LIDAR-Lite v3 are discussed. Diffuse reflective surfaces have a texture that causes the reflected light to disperse uniformly making them easy to detect because a portion of the reflected laser energy is nearly guaranteed to find its way back to the sensor. It is therefore desired that the majority of surface of the target is diffusely reflecting. The other type of surface if specular. Specular surfaces are smooth and tend to reflect energy with little to no dispersion. Therefore it can be difficult or even impossible to detect objects with such surface with the discussed sensor. The landing beams tend to reflect with little dispersion causing them to remain narrow and missing the sensor's receiver unless reflected directly back into it. A schematic comparison of behaviour of the laser beam when reflecting off of different types of surface is in Fig. 4.4.



(a) : Diffuse reflective surface. **(b) :** Specular surface.

Figure 4.4: Behaviour of laser beams reflecting off of surfaces with different reflective characteristics [24].

4.2 Complete Assembled Distance Measuring Mechanism

In this section, the completed and physically assembled prototype of the measuring mechanism is described. The prototype can be seen in Fig. 4.5.

4.2.1 Mechanism Parameters

The summary of physical parameters such as weight or size is provided in this subsection. Due to the fact that the mechanism is to be used in aerial applications, it places demands on the compact size and low weight also played a role when selecting each of the aforementioned components.

The mechanism dimensions and its weight are in Table 4.1. With length being the dimension in an imaginary x-axis, height in z and width in y-axis whose orientation and directions are the same as the ones of the UAV body coordinate system in Fig. 3.1.

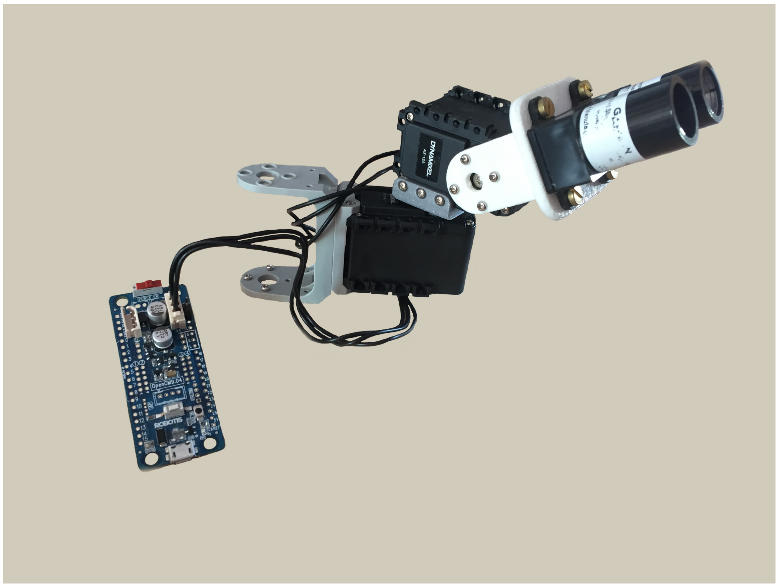


Figure 4.5: Assembled distance measuring mechanism with the OpenCM9.04-C microcontroller.

Table 4.1: Values of physical properties of the proposed mechanism.

Parameter	**Value**
length	125 mm
width	49 mm
height	80 mm
weight	158 g

The dimensions were measured when $[\phi \;\; \psi]^T = [0 \;\; 0]^T$, in other words, the mechanism was fully stretched.

Chapter 5

Software Driver Development

After having completed the hardware part of the project and thus having a mechanism that is capable of a distance retrieval, the next step is to develop a software driver that makes up the core of the proposed UVDAR extension. This chapter describes such a driver together with the programs running on the OpenCM9.04-C and Arduino Mega boards. It also gives a brief introduction into the used software interface of Robot Operating System (ROS).

5.1 Robot Operating System

Robot Operating System (ROS) is an open-source meta-operating system running on Unix-based platforms. It provides a widely used framework for writing robot software. ROS has numerous functionalities implemented, such as message transfer between running processes, low-level device control, package management and more [8]. Furthermore, ROS itself offers several powerful tools such as 3D visualization tool RViz[10], or rqt[11] which is a framework implementing various GUI tools that are useful for debugging. For more information on RViz see Section 6.2.

A crucial term when talking about ROS software is a package. Software in ROS is organized in packages and every package contains all the necessary

[10]http://wiki.ros.org/rviz
[11]http://wiki.ros.org/rqt

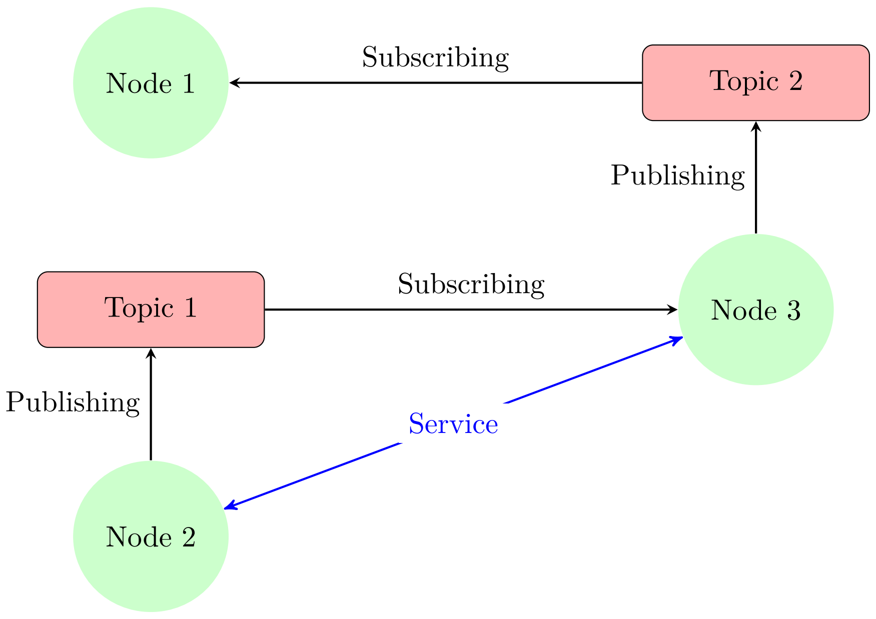


Figure 5.1: Example ROS communication diagram between multiple nodes.

files that are needed for the given program. Basic communication structure between multiple ROS programs is as follows. A running process is called a node. Nodes are connected via channels named topics, used for communication. A node can either publish data to the topic or subscribe the topic and receive data that way. Publishing is a process of uploading or posting data into the communication stream–topic. Subscribing is when the node reads the existing data from a topic. Another means of communication is provided by services. Services offer only one-to-one communication of type request/reply, but are also frequently used in a distributed system. One node can be connected to multiple topics and thus communicate with multiple other nodes. Each topic has a predefined message type that it transmits. Number of such message types is already defined, such as for common integers and floating point numbers, but also for messages representing sensor outputs like distance or even camera images. ROS also does not lack the option for the user to define his own message type that is most suitable for his application. A diagram of the communication between multiple ROS nodes can be seen in Fig. 5.1.

5.2 Main Control Node

In this section, the main control program is described. This program runs on the control unit of the UAV and its main task is to issue commands to the OpenCM9.04-C board and to make estimations of target position based on the measured distance.

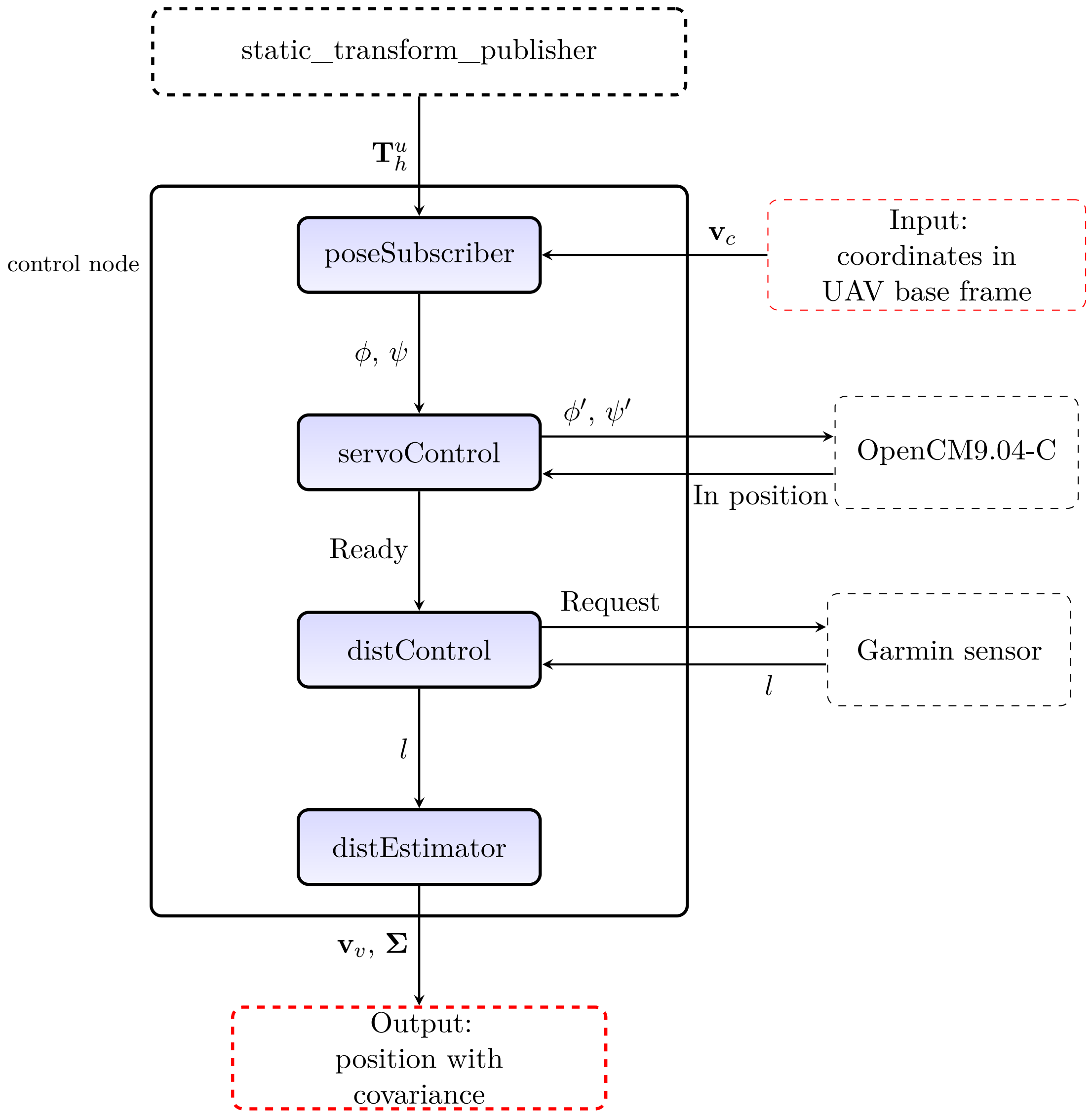


Figure 5.2: Data flow diagram of the control node.

The aforementioned task can be broken down into four simpler subtasks, each being taken care of by one C++ program, namely ’poseSubscriber’, ’servoControl’, ’distControl’ and ’distEstimator’. Despite being four different executables, all are enclosed by one ROS node. A data flow diagram of the ROS node is in Fig. 5.2 and a further explanation of each program with its tasks follows.

As mentioned in Section 3.4, the input into the UVDAR extension system is a 3D position of the target in the UAV base frame, which is subscribed by ’poseSubscriber’. This program first performs the needed coordinate transformations of the input position and then calculates ϕ and ψ using equations (3.8) and (3.9). The transformations are performed with help of the ROS `tf2` package, for detailed description of the package see [25]. This package contains useful tools for managing spatial transforms, such as the `static_transform_publisher`. The `static_transform_publisher` can be used for transforms that do not change over time, and thus is applicable for transforming the input position into the horizontal rotor frame, since the position of the horizontal servo is known beforehand. In order to perform a transform into the vertical rotor frame, the same procedure can not be followed, since its position is dependent on the angle ϕ, but an instance of `geometry_msgs::TransformStamped` is created. After the coordinate transformations are performed and the angles are calculated, ϕ and ψ are sent to 'servoControl', and thus another stage of the control program begins.

The 'servoControl' program takes care of all the communication with the OpenCM9.04-C board. The received values of ϕ and ψ are first recalculated to unitless absolute servo positions ϕ' and ψ', which are then sent to the OpenCM9.04-C board. After that it waits for an "in position" message from the microcontroller. Such a message signalls that the positioning mechanism is in position and the system is ready for distance retrieval. The program running on the OpenCM9.04-C board is described in 5.3. Upon receiving the anticipated message, the 'servoControl' requests distance measurement and passes control to the third program in order–'distControl'.

The main task of 'distControl' is to operate the Garmin sensor. After gaining control, the 'distControl' transmits message to the Garmin distance sensor signaling that a value of measured distance is required. The sensor answers with the measured distance. In the stage of ground testing, this communication was arranged by the Arduino Mega 2560, as is described in 5.4, but after mounting the extension onto the UAV, this procedure is able to run directly. The 'distControl' reads the value and sends measured distance to 'distEstimator' for further processing.

The last program–'distEstimator' has two main tasks. First, it creates a pose estimation $\mathbf{v}_v$ from the measured distance, as was described in Section 3.5. Then it calculates the covariance matrix $\mathbf{\Sigma}$ for the measurement and publishes both to the output topic as a `geometry_msgs/PoseWithCovariance` message.

5.3 OpenCM9.04-C Program

The program running on the OpenCM9.04-C board is described throughout this section. Its main function is to move with the servos, based on the commands coming from the main control node described in 5.2. The program described here also processes feedback from the servos.

As was mentioned in 4.1.1 the code was written in C programming language with the help of the Arduino IDE. For simplified communication with the AX-12A the DynamixelWorkbench.h[12] library was used. It offers an easy way of communication with different Dynamixel products using predefined library functions. An example use of the DynamixelWorkbench library for establishing connection to one servo and setting it to Joint Mode can be seen in the following Listing 5.1.

```
        DynamixelWorkbench dxl_wb;

        dxl_wb.init(DEVICE_NAME, DXL_BAUDRATE, &log);
        dxl_wb.ping(first_dxl_id, &model_number_one, &log);
        dxl_wb.jointMode(first_dxl_id, 0, 0, &log);
```

Listing 5.1: Setting up AX-12A communication using DynamixelWorkbench.h.

After having both of the servos set up, the program begins its main control loop. There, it first waits for new commands to be received. Upon receiving the new angles, the program checks whether they are within given boundaries that correspond to angles from the closed interval $\langle -\frac{\pi}{2}, \frac{\pi}{2} \rangle$. If it is not the case, the angles are replaced with the closest boundary and then sent to the servos, again taking advantage of the DynamixelWorkbench library and its functions. Otherwise they are sent to the servos directly. This is demonstrated in the Listing 5.2.

```
        dxl_wb.goalPosition(FIRST_DXL, (int32_t)goal_pos_hor);
        dxl_wb.goalPosition(SECOND_DXL, (int32_t)goal_pos_ver);
```

Listing 5.2: Sending desired positions to both servos.

Then the program waits until both of the servos are in position. That is done by reading present angles of each servo and comparing them to the set point. Since it is not always reliable to expect the present position to meet the set point exactly a small margin of ± 1 position is allowed. The code snippet with this waiting loop is in Listing 5.3.

```
while((abs(goal_pos_ver-pres_pos_ver)>1) || (abs(goal_pos_hor-pres_pos_hor)>1))
{
  dxl_wb.itemRead(FIRST_DXL, "Present_Position", &pres_pos_hor);
  dxl_wb.itemRead(SECOND_DXL, "Present_Position", &pres_pos_ver);
}
```

Listing 5.3: Loop that reads the present positions and compares them to given set point.

[12]http://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_workbench/

When both servos are in given positions, a message signaling that is sent back to the control unit running the main ROS node, described in Section 5.2.

5.4 Arduino Mega 2560 Program

Before the whole prototype is mounted onto an UAV and sensor is connected directly to its control unit, the reading of the data from the rangefinder has to be done externally. During the testing phase, the sensor was connected to an Arduino MEGA 2560. That was, with help of a simple program, able to read the output values and send them via serial line to the computer. The program used LIDARLite.h[13] library that offers a straightforward means for obtaining measured distance, see Listing 5.4. It first establishes connection with the sensor and in the second line the reading itself can be seen.

```
lidarLite.begin(0,true);
int16_t garmin_dist = lidarLite.distance();
```

Listing 5.4: Demonstration of communication with Garmin sensor using LIDARLite.h.

[13]https://www.arduinolibraries.info/ libraries/lidar-lite

Chapter 6

Simulation

Before every new device or system can be tested in real outdoor conditions it is essential to know that it behaves exactly the way it was intended. Therefore, each development procedure involves a simulation phase, where the system is tested in artificially created environment without the risk of the system being damaged or destroyed, should a malfunction occur. This chapter describes such simulation and debugging phase. Firstly, the used software tools such as the Gazebo[14] simulator and ROS visualization tool RViz are described, followed by a description of created model of the developed distance measuring system. Lastly, the performed experiments and tests in the simulator are introduced together with their results which are discussed in the last subsection.

6.1 Gazebo Simulator

Gazebo is an open-source 3D realistic robotics simulator. It is widely used for many robot applications and allows for simulations of robot activity in complex indoor and outdoor environments. It also offers the user the option to integrate realistic sensors and actuators that are vital for every robotic application. Gazebo can use multiple powerful physics engines, offering the user realistic simulations. Numerous tutorials and example projects exist, making the beginnings with Gazebo easier and allowing the user to advance to making his own worlds and simulating his own robots. Several ROS packages

[14]http://gazebosim.org/

can be used to provide the necessary interfaces to simulate a robot in Gazebo using ROS messages, making Gazebo a natural choice as a simulator to use alongside ROS. The Gazebo GUI together with a running simulation can be seen in Fig. 6.1.

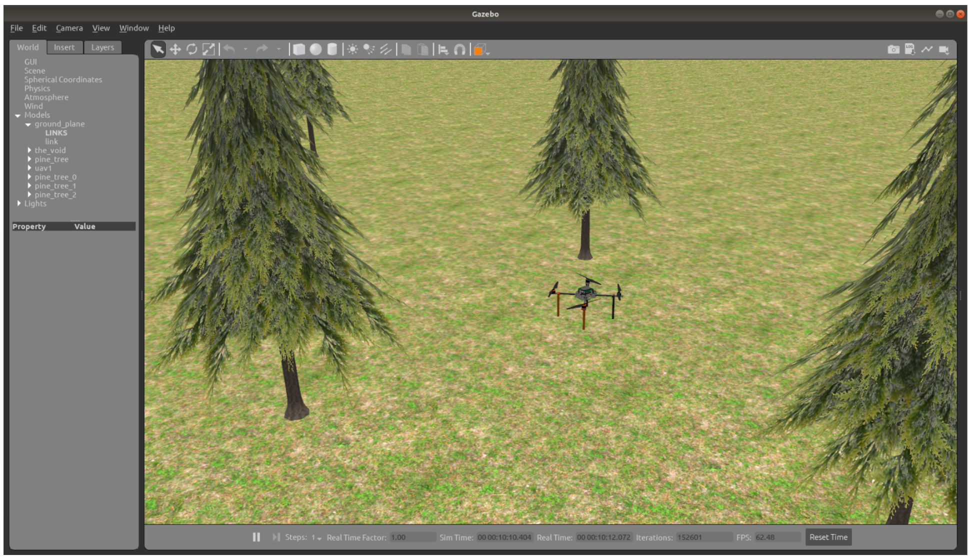
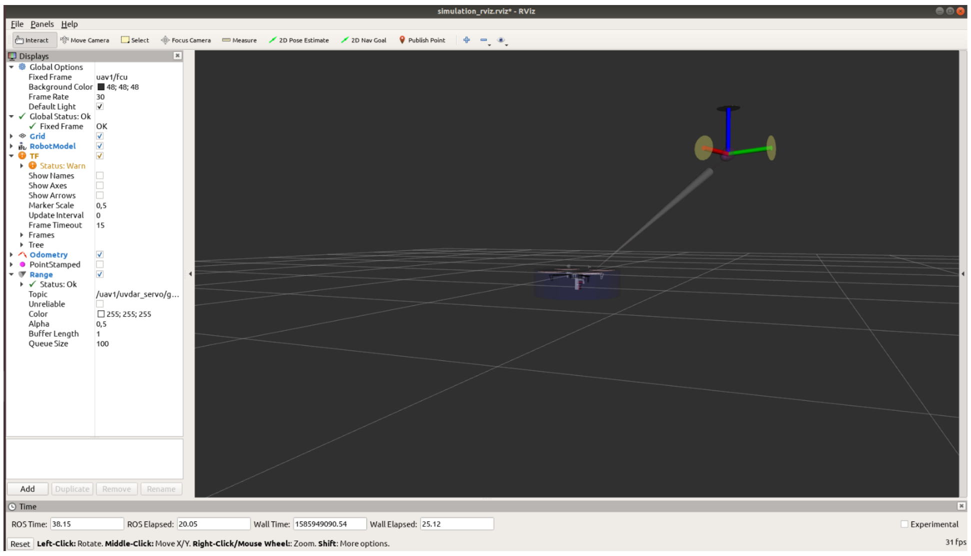


Figure 6.1: Gazebo GUI with running simulation.

6.2 3D Visualization Tool (RViz)

Another software that was used during the debugging phase was the 3D visualization tool for ROS RViz. Its main purpose is to show robot models and published data, such as position, camera images, range and many others in 3D, allowing for them to to be visually validated. Robot models that can be displayed with RViz need to be described in Unified Description Format[15] (URDF) which is then translated and expressed as a 3D model that can later be used for simulation or control [8]. Due to the fact that RViz can visualize data from ROS directly, it is a useful tool not only for validation of basic functions and correctness of the robot design. A view of the RViz GUI with a visualization of the proposed UVDAR extension consisting of an UAV robot model, target odometry and a laser cone, representing the measured distance is in Fig 6.2.

[15]http://wiki.ros.org/urdf



```
Global Options
  Fixed Frame         uav1/fcu
  Background Color    ■ 48; 48; 48
  Frame Rate          30
  Default Light       ✓
✓ Global Status: Ok
  ✓ Fixed Frame       OK
Grid                  ✓
RobotModel            ✓
TF                    ✓
  Status: Warn
  Show Names          ☐
  Show Axes           ☐
  Show Arrows         ☐
  Marker Scale        0,5
  Update Interval     0
  Frame Timeout       15
  Frames
  Tree
Odometry              ✓
PointStamped          ☐
Range                 ✓
  ✓ Status: Ok
  Topic               /uav1/uvdar_servo/g...
  Unreliable          ☐
  Color               ☐ 255; 255; 255
  Alpha               0,5
  Buffer Length       1
  Queue Size          100

ROS Time: 38.15   ROS Elapsed: 20.05   Wall Time: 1585949090.54   Wall Elapsed: 25.12   ☐ Experimental
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
```

Figure 6.2: Visualization with RViz.

6.3 UVDAR Extension Model for Gazebo

In order to test the proposed extension in simulation, a working model needed to be created. This subsection deals with such model and describes it. The manipulator and the mounted distance sensor models were created using Xacro, which is an XML macro language that allows for complex robot models to be parametrized and, in general, to simplify URDF files. The completed model file consists of multiple parts which are described in the following subsections.

6.3.1 The Xacro File

First, the description of the basic kinematic structure of the proposed system together with its key components is required. In order to do that, a link element for each part needs to be defined. Link is a structure used for describing the individual components of the robot model. For the model to properly work with Gazebo there are three compulsory tags for each link, namely the visual, collision and inertia tag. The visual tag is used for describing the appearance of the component. Collision tag defines the area of physical collision that the part would normally have. Last but not least, the inertia tag contains the 3×3 rotational inertia matrix of the component. The definition of the horizontal servo using XML can be seen in Listing 6.1

```
<link name="horizontal_servo">
  <visual>
    <geometry>
      <box size="${servo_width} ${servo_length} ${servo_height}">
    </geometry>
    <origin rpy="0 ${pi/2} 0" xyz="${servo_height/2} 0 0"/>
  </visual>
  <collision>
    <geometry>
      <box size="${servo_width} ${servo_length} ${servo_height}">
    </geometry>
    <origin rpy="0 ${pi/2} 0" xyz="${servo_height/2} 0 0"/>
  </collision>
  <inertial>
    <mass value="${mass}"/>
    <inertia
    ixx="${(mass/12)*(servo_length*servo_length+servo_height*servo_height)}"
    ixy="0"
    iyy="${(mass/12)*(servo_width*servo_width+servo_height*servo_height)}"
    iyz="0"
    izz="${(mass/12)*(servo_width*servo_width+servo_length*servo_length)}"
    ixz="0"
      />
  </inertial>
</link>
```

Listing 6.1: Definition of the horizontal servo body in Xacro format.

Multiple features and advantages of the Xacro format are apparent. It supports not only mathematical operations but also variables, parameters and conditional blocks.

Individual links are connected by joints. Various joint types are supported, such as revolute, continuous, prismatic, fixed, floating or planar, however, only fixed and revolute joints are used in this instance. Each joint is specified by two connected links and several other parameters according to its type, as can be seen in Listing 6.2. For the links used in this work they are lower and upper angle bound, its velocity and effort.

```
<joint name="horizontal_rotor_servo_joint" type="revolute">
  <axis rpy="0 0 0" xyz="0 0 1"/>
  <parent link="horizontal_servo"/>
  <child link="horizontal_rotor"/>
  <origin rpy="0 0 0" xyz="${rotor_x_offset} 0 0"/>
  <limit effort="100" velocity="6.28" lower="−${pi/2}" upper="${pi/2}"/>
</joint>
```

Listing 6.2: Definition of a revolute joint.

6.3.2 Transmission Elements and the Control Plugin

Another step in creating a functional model for Gazebo is to set up controllers to actuate the joints. The proper ROS interface for control of the actuators and joints is created using the `ros_control` package. This package uses a feedback mechanism with a PID controller to determine the given reference values based on the joint state data and a set point. To use the `ros_control`, a couple of additional elements need to be added into the model, namely the transmission elements, see Listing 6.3, and the `gazebo_ros_control` plugin. The transmission element is used to link actuators to joints. Specifically, it transforms effort/flow variables between actuators and joints such that the power remains constant [26]. Within the transmission tag it can be defined what hardware interface should the control plugin use, although only the effort interface is implemented at the time of writing [27]

```
<transmission name="${prefix}_tran">
  <type>transmission_interface/SimpleTransmission</type>
  <joint name="${prefix}_rotor_servo_joint">
    <hardwareInterface>
        hardware_interface/PositionJointInterface
    </hardwareInterface>
  </joint>
  <actuator name="${prefix}_motor">
    <hardwareInterface>
        hardware_interface/PositionJointInterface
    </hardwareInterface>
    <mechanicalReduction>1</mechanicalReduction>
  </actuator>
</transmission>
```

Listing 6.3: Code snippet that defines the transmission element.

In addition a Gazebo plugin needs to be added to the Xacro file. The `gazebo_ros_control` plugin parses the transmission tags and loads the controller manager and all the hardware interfaces. See Listing 6.4

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

Listing 6.4: Example code that adds the control plugin.

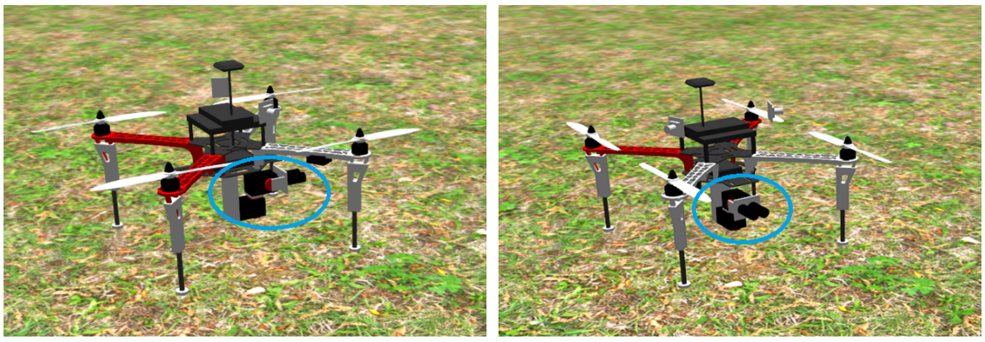
6.3.3 Garmin Sensor Model

The core of every Gazebo sensor model is a sensor plugin. This plugin defines the behavior and functions of the sensor. The plugin needs to be included into the model definition file as well, together with its parameters, as is shown in the Listing 6.5.

```xml
<plugin name="mrs_gazebo_rangefinder"
  filename="libmrs_gazebo_rangefinder_plugin.so">
  <gaussianNoise>${noise}</gaussianNoise>
  <alwaysOn>true</alwaysOn>
  <updateRate>${frame_rate}</updateRate>
  <topicName>${topic}</topicName>
  <frameName>${rangefinder_frame_name}</frameName>
  <fov>${fov}</fov>
  <radiation>radiation</radiation>
  <parentFrameName>${parent_frame_name}</parentFrameName>
  <x>${x}</x>
  <y>${y}</y>
  <z>${z}</z>
  <roll>${roll}</roll> <pitch>${pitch}</pitch> <yaw>${yaw}</yaw>
</plugin>
```

Listing 6.5: Inclusion of Garmin plugin in Xacro file.

Besides the plugin, a sensor link needs to be defined. In the link tag, there are visual and physical properties of the sensor together with its position in the model. A functioning Garmin sensor plugin that is in the MRS plugin database was used and only the visual link was slightly altered so it better corresponds to reality. A complete Gazebo model of the proposed UVDAR extension mounted onto a quadrotor, can be seen in Figs. 6.3a and 6.3b.

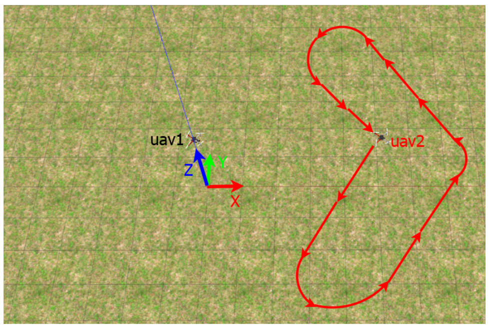


(a) : Side view. **(b) :** Front view.

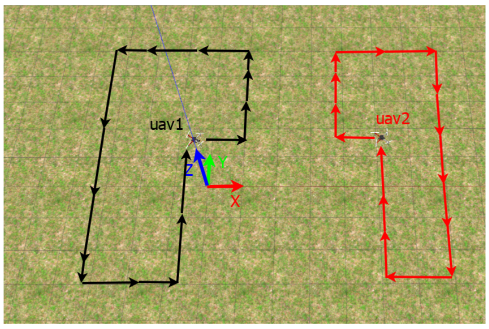
Figure 6.3: Complete model of the positioning mechanism with Garmin distance sensor in Gazebo. Marked with a blue ellipse.

6.4 Experiments in Simulator

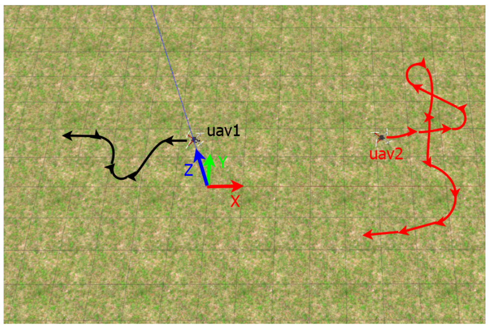
In this section, the performed experiments in the Gazebo simulator together with their results are presented. Four experiments were conducted to examine the influence of different flight dynamics on the performance of the proposed extension. In these experiments, only two UAVs were considered, one carrying the UVDAR extension, named 'uav1', and the other one–'uav2', serving as the observed target. The flight trajectories were chosen from the simplest, where the UAV carrying the extension does not move at all. Followed by trajectories where the relative position of the UAVs changes only slightly or the trajectories are quite simple and smooth. In the last experiment, both UAVs were traveling along complex, nearly chaotic, trajectories. For visualisation of each trajectory see Fig. 6.4.



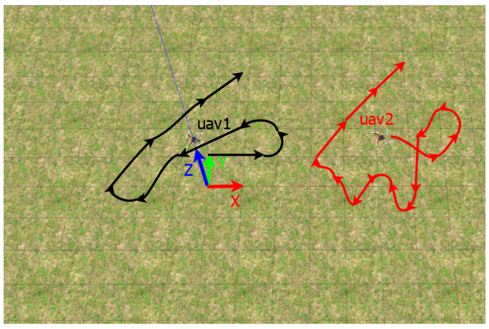
(a) : Trajectories for experiment 1.



(b) : Trajectories for experiment 2.



(c) : Trajectories for experiment 3.



(d) : Trajectories for experiment 4.

Figure 6.4: Flown trajectories for each experiment.

A working title of the proposed extension is 'uvdar servo' and that is also how it is denoted in the legends of each graph. The results of each experimental flight are presented in the form of a graph for each estimated relative coordinate. In these graphs, there is a comparison between the ground truth position, which was obtained from the 'uav2/ground _truth' topic published by Gazebo and the estimated positions both with and without employing the proposed extension. The absolute errors, $|\Delta x|$, $|\Delta y|$ and $|\Delta z|$, between the estimated and ground truth values of each coordinate are also graphically expressed in form of an another graph for each experiment. To quantify the performance of the localization systems, the average Euclidean distance between the ground truth and the estimated position is calculated. This gives another means of evaluation for the proposed extension other than just graphical intuition. The distance corresponding to estimates using the 'uvdar servo' is denoted by $\overline{d_{\mathrm{us}}}$, whereas for the UVDAR system it is just $\overline{d}$.

6.4.1 Experiment 1

In the first experiment, both UAVs were flying at equal altitude. The rangefinder was carried by 'uav1' and the trajectory of 'uav2' can be seen in Fig. 6.4a. The 'uav1' was stationary in the air and behaving solely as an observer. The comparison between the ground truth and the estimated relative positions and the graph of absolute errors for each coordinate are in Fig. 6.5. For the extended UVDAR, the average distance between the estimated and real position was $\overline{d_{\mathrm{us}}} = 0.451$ m, whereas for the system relying purely on the UVDAR system $\overline{d} = 1.057$ m.

6.4.2 Experiment 2

In the second experimental flight, both UAVs were already flying along trajectories in Fig. 6.4b. The altitudes of both UAVs were not equal any more, but were changing throughout the flight. Despite, that the trajectories were designed so that the relative y coordinate of both UAVs does not differ significantly. The results are summarized in Fig. 6.6 and the calculated deviations are $\overline{d_{\mathrm{us}}} = 0.527$ m and $\overline{d} = 1.316$ m.

6.4.3 Experiment 3

This experiment tested how precisely can the system track a fairly complex trajectory, while the 'uav1', with the distance sensor onboard, was traveling slowly along a smooth trajectory, see Fig. 6.4c, despite the fact that their relative altitude was changing. The results are visualized in Fig. 6.7 with average deviations of $\overline{d_{\mathrm{us}}} = 0.857$ m and $\overline{d} = 1.228$ m.

6.4.4 Experiment 4

In the last experiment, both UAVs were traveling along complex trajectories visualized in Fig. 6.4d. While flying at equal altitudes both trajectories were rich with numerous turns and sudden changes of direction which showed to be a bigger challenge for the proposed extension than controlled changes of relative altitude. The calculated average deviations in terms of distance are $\overline{d_{\mathrm{us}}} = 0.605$ m and $\overline{d} = 0.965$ m and the graphical representation of the results is in Fig. 6.8.

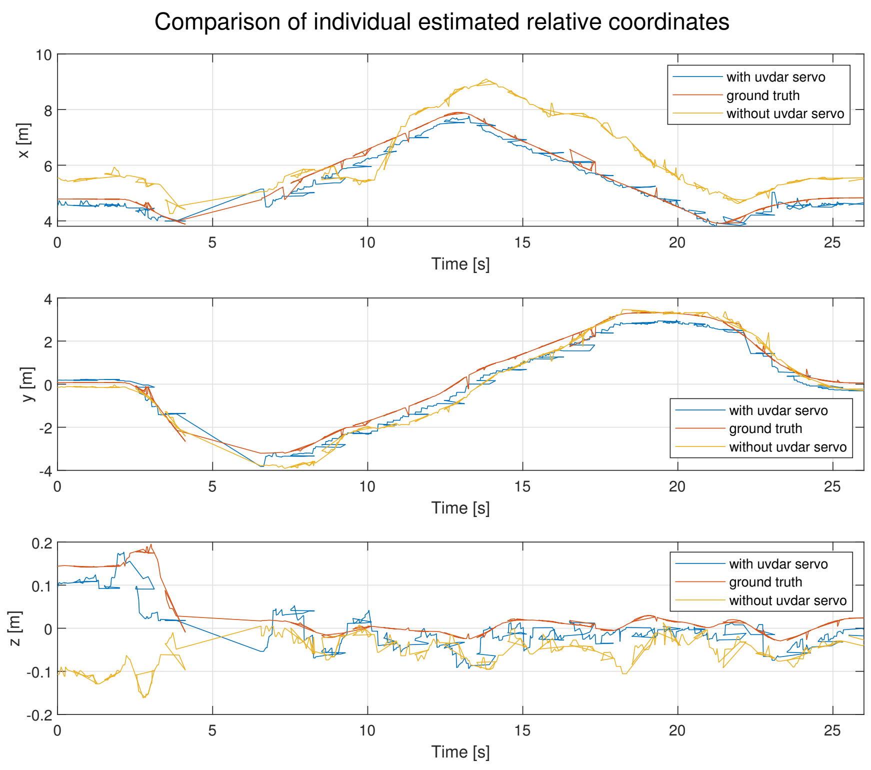
6.4.5 Discussion

The conducted experiments presented in the previous subsections have shown that employing the proposed extension in order to make the estimated positions more precise is justified. The estimations made by the extended UVDAR system proved to be not only comparable to the UVDAR system as it is but in multiple aspects and estimations even more precise, mostly in the estimations of the x coordinate. This can be seen when comparing the absolute errors of distinct coordinates for each experiment. For instance, while in Fig. 6.5b the values of $|\Delta x|$ are significantly lower for the extended UVDAR, the errors in estimates of y are comparable for both localization systems. This trend can be observed throughout all of the performed experiments. The notable difference in the precision of estimated values of x, compared with other coordinates, may be caused by the nature of the tested trajectories because it was the relative x coordinate that was changing the most and thus having the biggest impact on the relative distance of both UAVs. The extended UVDAR had the means to precisely measure that distance which made the estimation in that aspect better.

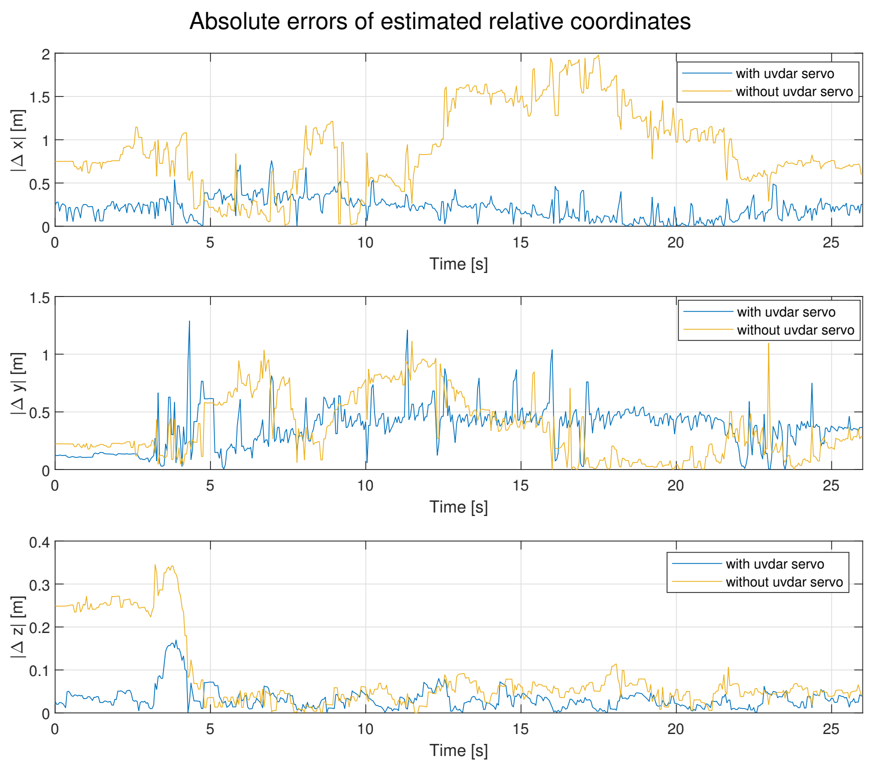
What further speaks to support the claim of a successful development of the extension, is the calculated average deviation between the estimated and real position. The fact, that throughout all of the conducted experiments, regardless of relative distances between the UAVs, it did not exceed the value of 1 m, can be considered as a further proof of that. Furthermore, when the flown trajectories were simple, as in experiments 6.4.1 and 6.4.2, the error of estimations made with the 'uvdar servo' extension showed to be less then the half of the deviation calculated for the system relying only on the unextended UVDAR.

However, it is safe to say that the performance is quite heavily based on the flight dynamics. While in experiments 6.4.1 and 6.4.2, where the flown trajectories of both UAVs were simple, the estimated positions by the proposed system corresponded with the ground truth closely, in the other two experiments where the trajectories were more complex, even these estimations failed to determine the real position with such precision. The takeaway from the experiments 6.4.3 and 6.4.4 is that the reliability and precision depends more on the trajectory of the UAV which carries the extension, than the observed one. This can be seen when comparing the values of absolute errors $|\Delta y|$ for both experiments. While with experiment 6.4.3, the estimations made by the 'uvdar servo' are slightly better than those made by the unextended UVDAR, the data provided by experiment 6.4.4 shows, that the extension did not make the estimations more precise by a larger margin, if at all. See Figs. 6.7b and 6.8b for a comparison of $|\Delta y|$ for both experiments. The apparent errors in estimations of z for both of these experiments were most likely caused by the tilt of the 'uav1' while changing direction, which resulted in shifting the position of 'uav2' in the camera frame. This analysis is supported by the fact that said errors are not constant but appear only at certain points in time that correspond to changes in trajectory. In this application the tilt of the UAV was not regarded, since it would need some form of disturbance control mechanism, which is out of the scope of this work. However it presents a possible motivation for future work.

The experiments have shown that the development of the UVDAR extension was successful. The aim was to enhance the performance of the UVDAR system in terms of distance measuring which resulted in making the target position estimations more precise in general. While being more than satisfyingly precise in flights along smooth and simple trajectories the system performed sufficiently even for chaotic and very dynamic trajectories, making it a system capable of real flight testing and deployment.

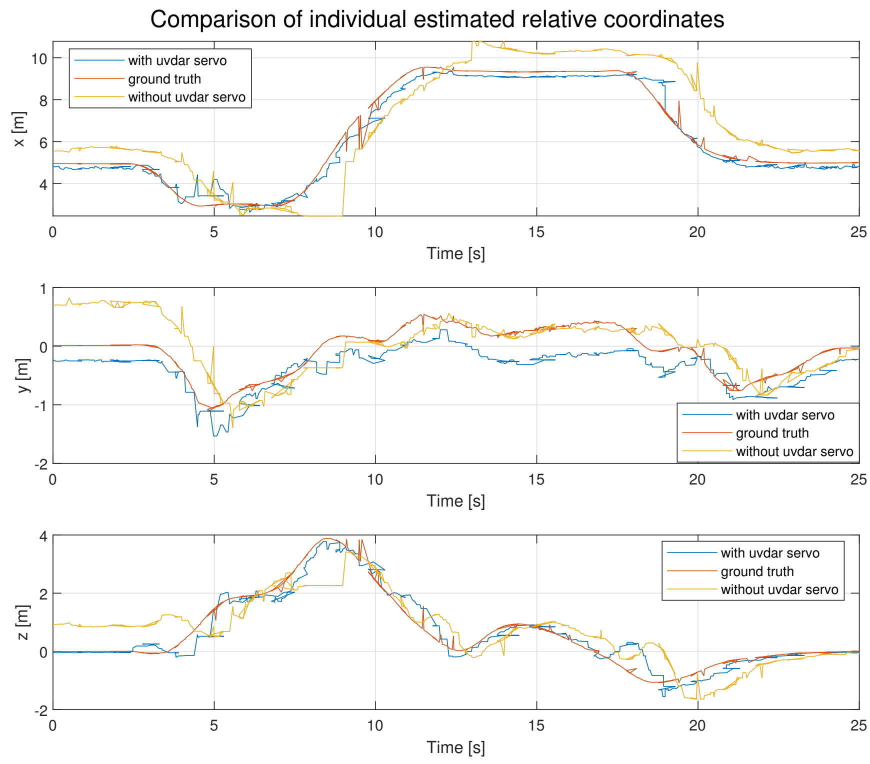


(a) : Visualization of estimated coordinates for experiment 1.

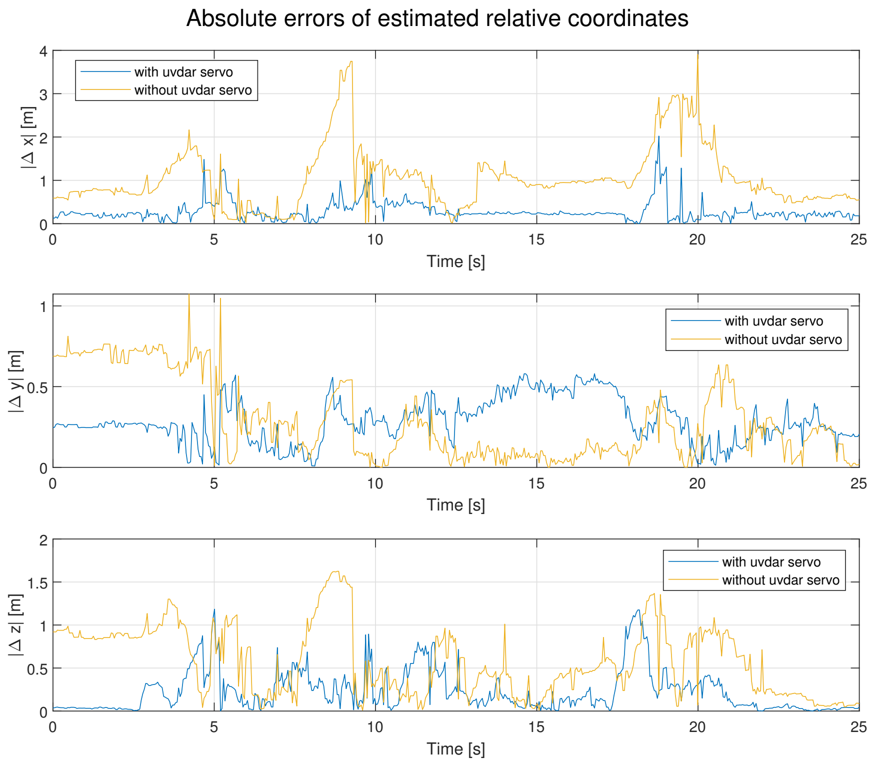


(b) : Comparison of absolute errors of each coordinate for experiment 1.

Figure 6.5: Measured data for experiment 1.

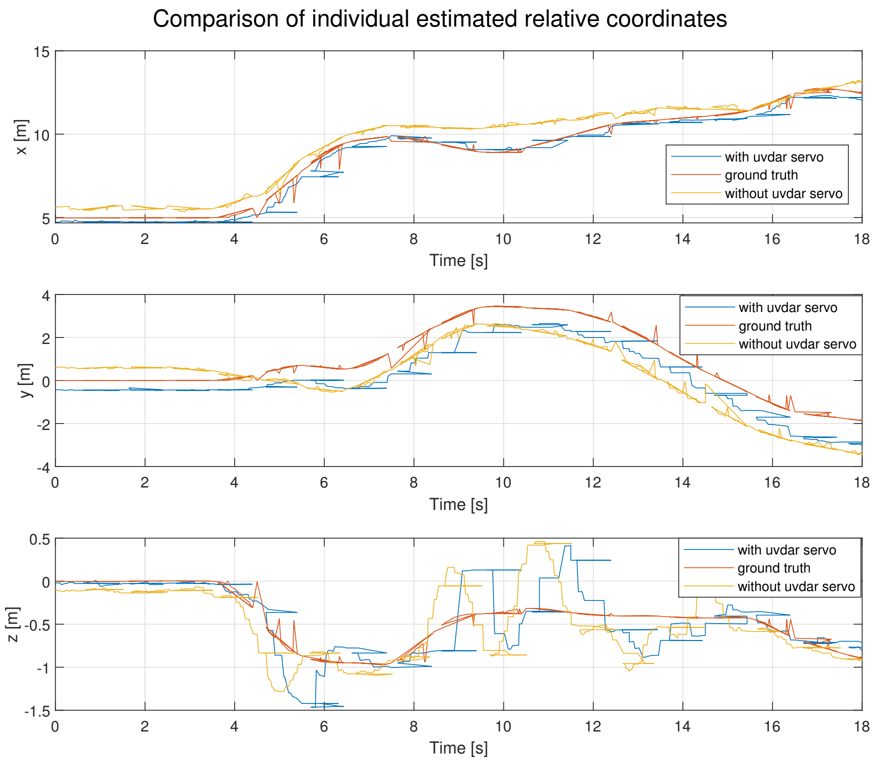


(a) : Visualization of estimated coordinates for experiment 2.

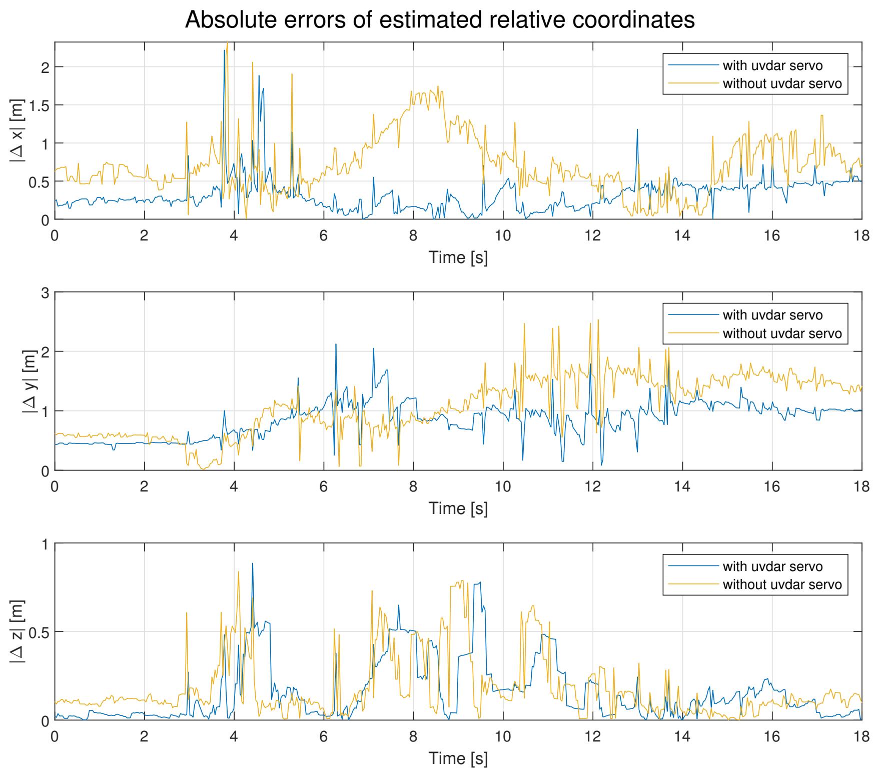


(b) : Comparison of absolute errors of each coordinate for experiment 2.

Figure 6.6: Measured data for experiment 2.

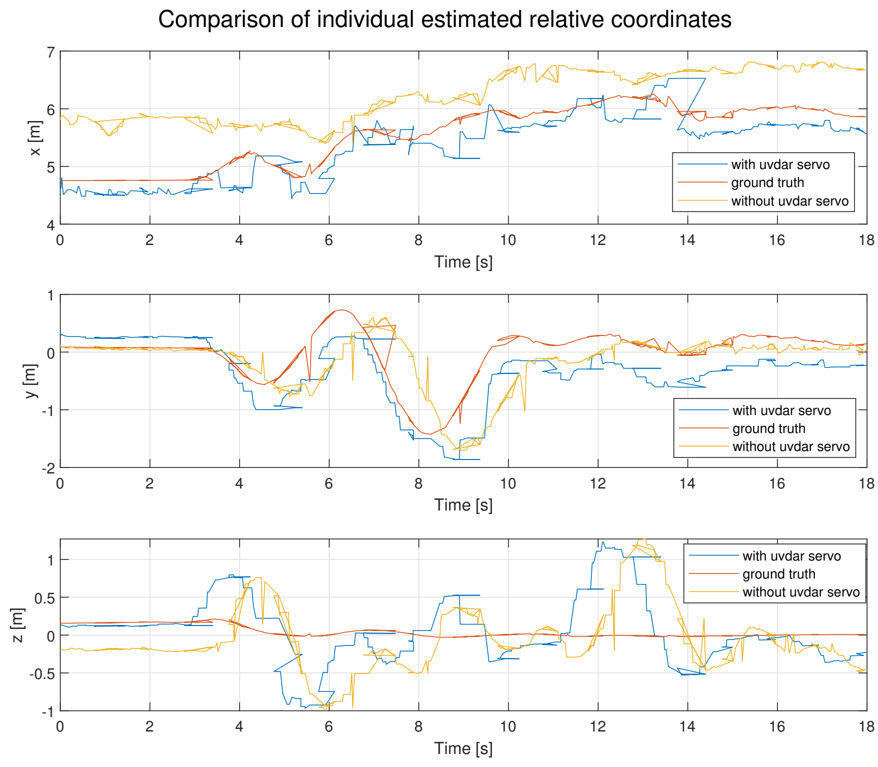


(a) : Visualization of estimated coordinates for experiment 3.

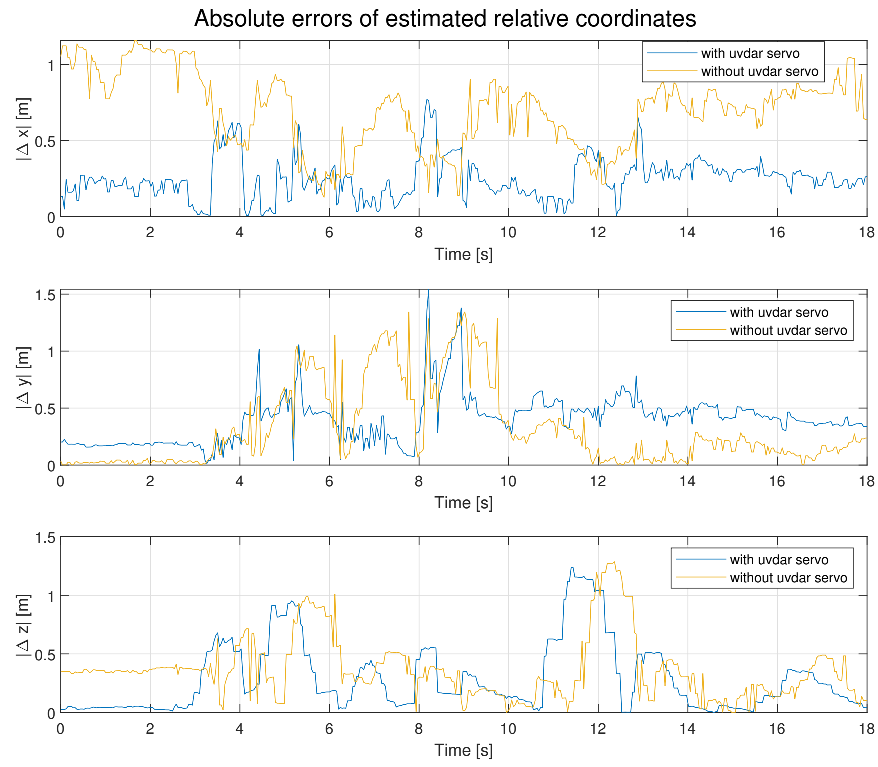


(b) : Comparison of absolute errors of each coordinate for experiment 3.

Figure 6.7: Measured data for experiment 3.



(a) : Visualization of estimated coordinates for experiment 4.



(b) : Comparison of absolute errors of each coordinate for experiment 4.

Figure 6.8: Measured data for experiment 4.

Chapter 7

Conclusion

The aim of this thesis was to develop an extension of the UVDAR system for mutual relative localization that would use precise distance measurements in order to enhance the performance of the UVDAR system in terms of target position estimation.

In addressing the challenge, common, but high–performance, off–the–shelf components were used in order to assemble the prototype. The main reasoning behind the choice of each of the components was their weight, since they are used in aerial application, and their compatibility, so they could be easily connected together and controlled. Despite that, some compromises and improvised solutions had to be found during the development phase, such as the inclusion of the extra Arduino Mega 2560, which enabled the communication with the Garmin sensor. In order to test the developed control programs, the system had to be tested in simulation and therefore a model of the proposed UVDAR extension was created. Initially it was intended, that the development would be rounded off by a real flight experiment. This was unfortunately impossible, due to the current pandemic situation. However the tests performed in the Gazebo simulator show promising results, leaving the doors open for proceeding to the real flight experiments as soon as the situation allows it.

The main contribution of the work presented here lies in developing a prototype of the proposed UVDAR extension together with all the needed control software. This provides a solid ground for future work that concerns mainly real flight experiments and validation of the proposed extension in real conditions. Furthermore, the system can be easily modified for flights with more than just one target. This would provide the UAV with a system that can precisely estimate position of multiple surrounding targets even at large distances, on top of being reliable and easily embeddable. All of the above is vital for any application where some form of UAV cooperation is required.

Bibliography

[1] A. Ben-David. IAI reveals largest Israeli UAV. June 2007.

[2] L. Schenato, X. Deng, and S. Sastry. Flight control system for a micromechanical flying insect: architecture and implementation. In *Proceedings of IEEE Conference on Robotics and Automation (ICRA 01)*, volume 2, pages 1641–1646, Taipei Taiwan, September 2001.

[3] Corporate Author. *U.S. Army Unmanned Aircraft Systems Roadmap 2010-2035: Eyes of the Army*. United States. Department of the Army, April 2010.

[4] A. Yol, B. Delabarre, A. Dame, J. Dartois, and E. Marchand. Vision-based absolute localization for unmanned aerial vehicles. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3429–3434, September 2014.

[5] *Micro Aerial Vehicles - platforms*, 2020. Multi-robot-Systems Group website, Available at `http://mrs.felk.cvut.cz/research/micro-aerial-vehicles`.

[6] V. Walter, M. Saska, and A. Franchi. Fast mutual relative localization of UAVs using ultraviolet led markers. In *2018 International Conference of Unmanned Aircraft System (ICUAS 2018)*, 2018.

[7] Marco Cognetti, Paolo Stegagno, Antonio Franchi, and Giuseppe Oriolo. Two Measurement Scenarios for Anonymous Mutual Localization in Multi-UAV systems. *IFAC Proceedings Volumes*, 45(28):13–18, 2012. 2nd IFAC Workshop on Multivehicle Systems.

[8] Yoonseok Pyo, Hancheol Cho, Leon Jung, and Darby Lim. *ROS Robot Programming (English)*. ROBOTIS, December 2017.

[9] Aleksandr Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Auton. Robots*, 35(4):287–300, 2013.

[10] Gábor Vásárhelyi, Csaba Virágh, Gergo Somorjai, Norbert Tarcai, Tamás Szörényi, Tamás Nepusz, and Tamás Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. *CoRR*, abs/1402.3588, August 2014.

[11] M. Cognetti, P. Stegagno, A. Franchi, G. Oriolo, and H. H. Bülthoff. 3-D mutual localization with anonymous bearing measurements. In *2012 IEEE International Conference on Robotics and Automation*, pages 791–798, May 2012.

[12] L. Zhang, F. Deng, J. Chen, Y. Bi, S. K. Phang, X. Chen, and B. M. Chen. Vision-Based Target Three-Dimensional Geolocation Using Unmanned Aerial Vehicles. *IEEE Transactions on Industrial Electronics*, 65(10):8052–8061, October 2018.

[13] Paul Viola and William Wells. Alignment by Maximization of Mutual Information. *International Journal of Computer Vision*, 24:137–154, January 1997.

[14] J. Strader, Y. Gu, J. N. Gross, M. De Petrillo, and J. Hardy. Cooperative relative localization for moving UAVs with single link range measurements. In *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 336–343, April 2016.

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

[16] V. Walter, N. Staub, A. Franchi, and M. Saska. UVDAR System for Visual Relative Localization With Application to Leader–Follower Formations of Multirotor UAVs. *IEEE Robotics and Automation Letters*, 4(3):2637–2644, July 2019.

[17] Pavel Petráček and Martin Saska. Decentralized Aerial Swarms Using Vision-Based Mutual Localization. November 2018.

[18] Daniel Brandtner and Martin Saska. Coherent swarming of unmanned micro aerial vehicles with minimum computational and communication requirements. In *ECMR*, 2017.

[19] James Eddie Gentle. *Matrix Algebra*, chapter 5, pages 173 – 200. Springer International Publishing, June 2007.

[20] J. Z. Sasiadek and P. Hartana. Sensor data fusion using Kalman filter. In *Proceedings of the Third International Conference on Information Fusion*, volume 2, pages WED5/19–WED5/25 vol.2, 2000.

[21] Dan Simon. Kalman filtering. *Embedded systems programming*, 14(6):72–79, 2001.

[22] ROBOTIS. *OpenCM9.04-C e-manual*, 2020. ROBOTIS website, Available at `http://emanual.robotis.com/docs/en/parts/controller/opencm904/`.

[23] ROBOTIS. *Dynamixel AX-12A e-manual*, 2020. ROBOTIS website, Available at `http://emanual.robotis.com/docs/en/dxl/ax/ax-12a/`.

[24] Garmin. *LIDAR Lite v3 Operation Manual and Technical specifications*, 2016. Product datasheet, Available at `http://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf`.

[25] Foote and Tully. tf: The transform library. In *Technologies for Practical Robot Applications (TePRA), 2013 IEEE International Conference on*, Open-Source Software workshop, pages 1–6, April 2013.

[26] Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, and Enrique Fernández Perdomo. ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*, 2017.

[27] Open Source Robotic Foundation. *Tutorial: ROS Control*, 2014. Gazebo tutorial, Available at `http://gazebosim.org/tutorials/?tut=ros_control`.

Appendix A

List of Abbreviations

Table A.1 explains the abbreviations used throughout this thesis.

Table A.1: List of Abbreviations.

Abbreviation	Meaning
UAV	Unmanned Aerial Vehicle
MFI	Micromechanical Flying Insect
UVDAR	Ultraviolet Direction and Ranging
ROS	Robot Operating System
GNSS	Global Navigation Satellite System
DOF	degree of freedom
ARM	family of RISC architectures for processors
CPU	central processing unit
SRAM	semiconductor random–access memory
GPIO	general–purpose input/output
UART	Universal Asynchronous Receiver/Transmitter
JTAG	hardware communication interface (Joint TestAccess Group)
SWD	Serial Wire Debug
USB	Universal Serial Bus
SPI	serial communication interface (Serial Peripheral Interface)
I2C	serial computer bus
TTL	Transistor-transistor logic
IDE	Integrated development environment
RAM	random–access-memory
EEPROM	electrically erasable programmable read–only memory
XML	Extensible Markup Language
URDF	Unified Robot Description Format

Appendix B

CD Content

Table B.1 shows all the root directories on the attached CD with their content.

Table B.1: CD content.

Directory	Content
thesis	Bachelor's thesis in pdf format
thesis_src	LaTeX source codes
ROS_src	ROS control node source files
model	Xacro model of the extension for Gazebo
uc_src	OpenCM9.04-C and Arduino MEGA 2560 codes