TU WIEN Informatics

Coordinated control of ground and aerial vehicle during takeoff and landing

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Informatik

eingereicht von

Maximilian Engl, BSc
Matrikelnummer 11775811

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Mitwirkung: Univ.Ass. Dipl.-Ing. Andreas Brandstätter, BSc

Wien, 25. März 2024

Maximilian Engl	Radu Grosu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

Coordinated control of ground and aerial vehicle during takeoff and landing

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Computer Engineering

by

Maximilian Engl, BSc
Registration Number 11775811

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu
Assistance: Univ.Ass. Dipl.-Ing. Andreas Brandstätter, BSc

Vienna, 25th March, 2024

Maximilian Engl	Radu Grosu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

Maximilian Engl, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 25. März 2024

Maximilian Engl

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

In erster Linie möchte ich meiner Familie und im Besonderen meinen Eltern danken, die mir mein Studium überhaupt ermöglicht haben und mich immer und in allem unterstützt haben!

Ein besonders großes Dankeschön gilt meiner Freundin Alisa, die mir während des ganzen Studiums immer zur Seite gestanden ist und immer ein offenes Ohr für mich hatte.

Auch bedanke ich mich bei meinen Freunden, die mich stetig motivierten, aber auch für die nötige Ablenkung sorgten.

Abschließend möchte ich mich natürlich auch bei Univ.Ass. Dipl.-Ing. Andreas Brandstätter, BSc und Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu für die Betreuung meiner Arbeit und die fachliche Unterstützung bedanken. Ohne sie wäre meine Arbeit nicht möglich gewesen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

First and foremost, I would like to thank my family and especially my parents, who have made my studies possible and have always supported me in everything!

A particularly big thank you goes to my girlfriend Alisa, who always stood by my side throughout my studies and always had an open ear for me.

I would also like to thank my friends, who constantly motivated me, but also provided the necessary distractions.

Finally, I also want to thank Univ.Ass. Dipl.-Ing. Andreas Brandstätter, BSc and Univ.Prof. Dipl.-Ing. Dr.rer.nat. Radu Grosu for supervising my thesis and for their expertise and support. Without them, my thesis would not have been possible.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Die Zusammenarbeit zwischen unbemannten Luftfahrzeugen (englisch unmanned aerial vehicles, UAVs) und Bodenfahrzeugen kann vorteilhafte Eigenschaften für eine Vielzahl unterschiedlicher Aufgaben in der Robotik mit sich bringen. Entscheidend für ein solches kooperatives System ist die Durchführung erfolgreicher Start- und Landemanöver von UAVs auf sich bewegenden Bodenfahrzeugen. Besonders in diesen Phasen ist die Koordination zwischen UAV und Bodenfahrzeug entscheidend, und eine genaue relative Lokalisierung spielt dabei eine wichtige Rolle.

In dieser Arbeit stellen wir ein System vor, das es einem Quadrotor-UAV ermöglicht, autonom zu starten und auf einem sich bewegenden Bodenfahrzeug zu landen, indem wir sowohl ein System zur Verbesserung der Genauigkeit der UAV-Positionsschätzung unter Verwendung einer handelsüblichen Kamera als auch ein High-Level-Controller-Design vorschlagen, das die Landetrajektorie für das UAV generiert.

Die UAV-Positionsschätzung wird mit Hilfe von Bezugsmarken verbessert, die an der Unterseite des UAV angebracht sind und deren relative Position zu den Bodenfahrzeugen bestimmt wird, nachdem sie von der Kamera des Bodenfahrzeugs erfasst wurden. Der erweiterte Kalman-Filter des UAV fusioniert die Schätzungen mit Hilfe eines Noise-Modells, das wir in einer Simulationsumgebung abgeleitet haben.

Die generierten Landetrajektorien basieren auf Bèzier-Kurven, die standardmäßig kontinuierlich sind und deren Krümmung durch eine Reihe von Kontrollpunkten festgelegt werden kann. Basierend auf der aktuellen Position der UAV und der vorhergesagten zukünftigen Position des Bodenfahrzeugs wird die Landetrajektorie kontinuierlich generiert, mit dem sekundären Ziel, das relative Lokalisierungssystem durch eine Annäherung an das Bodenfahrzeug im Blickfeld der Kamera zu unterstützen.

Wir evaluieren die Lokalisierungsimplementierung quantitativ auf realer Hardware, um ihre Effektivität zu zeigen und die Leistung zu bewerten. Darüber hinaus testen wir den High-Level-Controller erfolgreich in einem Versuchsaufbau bestehend aus einer Crazyflie 2.1 Quadrotor-Drohne und einem Modell-Rennwagen mit befestigter Landeplattform. Wir sind in der Lage, zuverlässig auf dem stillstehenden Bodenfahrzeug zu landen. Mit der verwendeten Computerplattform kann der Controller nur mit einer Frequenz von 10 Hz arbeiten, wodurch das System zwar auch für sich bewegenden Landeplatformen funktioniert, dabei jedoch nicht immer zuverlässig ist.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Collaboration between Unmanned aerial vehicles (UAVs) and ground vehicle can bring beneficial properties for a wide range of different tasks in field robotics. Critical for such a collaborative system is the execution of successful takeoff and landing maneuvers of UAVs on moving ground vehicles. Especially in this phase, coordination between the UAV and the ground vehicle is crucial, and accurate relative localization plays a significant role in achieving it.

In this work, we therefore present a system that allows a quadrotor UAV to autonomously take off and land on a moving ground vehicle by proposing both a system to improve the accuracy of UAV position estimation using an off-the-shelf camera and a high-level controller design that generates the landing trajectory for the UAV.

The UAV's position estimation is improved with the help of fiducial markers attached to the bottom of the UAV, whose relative position to the ground vehicles is estimated after being detected by the camera on the ground vehicle. The UAV's Extended Kalman filter fuses the estimations using a noise model we derived in simulation.

The generated landing trajectories are based on Bèzier curves, which are smooth by design and whose curvature can be specified using a set of control points. Based on the current position of the UAV and the predicted future position of the ground vehicle, the landing trajectory is constantly generated with the secondary objective of helping the relative localization system by approaching the ground vehicle in the camera's view.

We quantitatively evaluate the localization implementation on real hardware, to show its effectiveness and asses the performance. We furthermore successfully test the high-level landing controller in an experimental setup consisting of a Crazyflie 2.1 quadrotor UAV and a model race car with attached landing platform. With the computer platform used, the controller can only operate at a frequency of 10 Hz, which means that although the system also works for moving landing platforms, it is not always reliable.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

1.1 Motivation & Problem Statement

Unmanned aerial vehicles (UAVs) are becoming increasingly popular for performing different tasks, ranging from remote monitoring and mapping to search and rescue (SAR) [39, 12]. Despite their versatile applications, several challenges must be overcome, such as short flight duration and low payload capacity.

A possible approach to tackle these difficulties is to combine a ground vehicle with an aerial vehicle, thus compensating the respective disadvantages. This way, a UAV can move as part of the ground vehicle and only deploy near the operation site. In addition, it can be ensured that there is always a proper landing platform within the range where the UAV can recharge. Furthermore, the ground vehicle is always within range, which means it can perform the more complex computational work and even take over the control of the aerial vehicle, thus reducing its computing hardware requirements.

Not only does the aerial vehicle benefit from the ground vehicle, but also the other way around. Suppose the path is obstructed or a reorientation is necessary. In that case, the aerial vehicle can get a better overview by providing a bird's eye view of the current situation. The aerial vehicle can search for alternate paths or reach locations inaccessible to the ground vehicle [32]. It also allows the aerial vehicle to be used as a radio relay and thus extend the range of an existing radio network [13].

1.2 Problem Statement

We seek to develop a system that allows a quadrotor UAV to autonomously take off and land on a moving ground vehicle. The goal of this master thesis is to design a controller that allows a UAV to take off and land autonomously on a moving vehicle. The controller can interface with the vehicle and the UAV to determine their best actions.

One or more localization systems are needed to determine the absolute and relative position of the involved vehicles. Due to hardware constraints, different ground and aerial vehicle localization systems have to be used. Therefore, the uncertainties of the various systems have to be considered. In addition, both systems are too inaccurate to land reliably, so additional relative localization might be used to improve the positioning of the UAV relative to the ground vehicle once they are close to each other. These systems need to be appropriately fused to achieve accurate and reliable localization.

To our knowledge, all relevant previous vision-based implementations use cameras attached to the UAV itself (see Section 2). However, this is not feasible since lightweight UAVs are used in this work. Therefore, we are investigating alternative approaches to solve this problem.

1.3 Structure

In the next chapter, related work is reviewed, and theoretical concepts, such as probabilistic theory, state space, and applications in localization, are introduced. The paper's core, the system design, is detailed in Chapter 3, and discusses the robotic platform's capabilities, feasibility, and the integration of relative position measurements and the landing controller's design. Chapter 4 provides an analysis of UAV positioning, landing target prediction, and overall landing performance. Finally, the paper concludes with a summary of the findings and offers an outlook on future research directions.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

State of the Art and Theoretic Concepts

2.1 Related Work

As take-off is much easier than landing, the literature mainly concerns the latter. The landing platform is usually assumed stationary [42, 48, 31], but research is also dedicated to moving platforms[46, 45, 2].

Rucco et al. [46] used a trajectory-tracking approach with known positions to find optimal rendezvous trajectories for ground vehicles and fixed-wing UAVs. A similar approach for quadrotor UAVs is pursued by Guatam et al. [20], where the authors define 3D guidance laws based on pure pursuit, line-of-sight, and pure proportional navigation.

Given that vision sensors are frequently part of a UAV's hardware configuration, they are often used as part of the landing system. Most approaches use markers on the landing platform, which are detected by image feature detectors (like SIFT [33] and SURF [6]). Using the known dimensions of the marker and the intrinsic camera parameters, it is then possible to estimate the position of the UAV relative to the landing pad, see [48, 2, 31].

Rodriguez-Ramos et al. [45] and Polvara et al. [42] combine vision-based landing with deep reinforcement learning to train the landing maneuver using simulation. This has the advantage that no high-resolution images are required, so it can be easily implemented with cheaper hardware. They achieved good results in the simulation. However, the evaluation on real hardware showed degraded performance, most likely because of a possible sim-to-real transfer gap.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.2 Probabilistic Theory

The following sections give an overview of the fundamentals of probabilistic theory and how it is used in robotics by summarizing the core concepts of [51], as they are fundamental for understanding state estimation.

Random variables hold values according to distinct principles and are used in probabilistic robotics to model quantities such as sensor measurements, controls, and system states of robots and environments. A random variable can be a scalar or a vector of scalar random variables.

Let X be a random variable and $\mathbf{x}$ be a specific value in its domain, then $p(X = \mathbf{x}) = p(\mathbf{x})$ gives us the probability that the random variable X has the value $\mathbf{x}$. Additionally, the following holds:

$$0 \leq p(\mathbf{x}) \leq 1 \quad \text{for all possible values of } \mathbf{x} \tag{2.1}$$

$$\sum_{\mathbf{x}} p(\mathbf{x}) = 1 \quad \text{(discrete case)} \tag{2.2}$$

$$\int_x p(\mathbf{x})\, d\mathbf{x} = 1 \quad \text{(continuous case)} \tag{2.3}$$

Given two random variables X and Y, $p(X = \mathbf{x}, Y = \mathbf{y}) = p(\mathbf{x}, \mathbf{y})$ gives the joint distribution of those two random variables. If X and Y are *independent*, i.e., the occurrence of one random variable does not influence the probability distribution of the other and vice versa, the following holds:

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) \cdot p(\mathbf{y}) \tag{2.4}$$

Suppose the value of Y is known to be $\mathbf{y}$, $p(X = \mathbf{x} | X = \mathbf{y}) = p(\mathbf{x}|\mathbf{y})$ gives the so called *conditional* probability of $\mathbf{x}$. If $p(\mathbf{y}) > 0$, it is defined as

$$p(\mathbf{x}|y) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{y})}. \tag{2.5}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Using the *theorem of total probability*, it is possible to calculate the probability of the first variable if the conditional probabilities, as well as the probabilities of the second variable, are known:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}|\mathbf{y})\ p(\mathbf{y}) \quad \text{(discrete case)} \tag{2.6}$$

$$p(\mathbf{x}) = \int_{y} p(\mathbf{x}|\mathbf{y})\ p(\mathbf{y})\ d\mathbf{y} \quad \text{(continuous case)} \tag{2.7}$$

By inserting (2.6) resp. (2.7) in (2.5), the *Bayes rule* can be formulated, also requiring $p(\mathbf{y}) > 0$:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})\ p(\mathbf{x})}{p(\mathbf{y})} = \frac{p(\mathbf{y}|\mathbf{x})\ p(\mathbf{x})}{\sum_{\mathbf{x}'} p(\mathbf{y}|\mathbf{x}')\ p(\mathbf{x}')} = \eta\ p(\mathbf{y}|\mathbf{x})\ p(\mathbf{x}) \quad \text{(discrete case)} \tag{2.8}$$

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})\ p(\mathbf{x})}{p(\mathbf{y})} = \frac{p(\mathbf{y}|\mathbf{x})\ p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x}')\ p(\mathbf{x}')\ d\mathbf{x}'} = \eta\ p(\mathbf{y}|\mathbf{x})\ p(\mathbf{x}) \quad \text{(continuous case)} \tag{2.9}$$

Since $p(\mathbf{y})^{-1}$ is independent of $\mathbf{x}$, it can be replaced by the *normalization factor* η, which ensures that the sum resp. integral over $p(\mathbf{x}|\mathbf{y})$ is 1.

The Bayes rule is fundamental in probabilistic robotics because it relates the conditionals probabilities $p(\mathbf{x}|\mathbf{y})$ with their *inverse* $p(\mathbf{y}|\mathbf{x})$ which is often easier to calculate or already defined in the system.

The *expected value* of a random variable X is given by

$$E[X] = \sum_{\mathbf{x}} \mathbf{x}\ p(\mathbf{x}) \quad \text{(discrete case)} \tag{2.10}$$

$$E[X] = \int \mathbf{x}\ p(\mathbf{x}) d\mathbf{x} \quad \text{(continuous case)} \tag{2.11}$$

The *covariance* indicates the squared expected deviation from the mean and is given by

$$Cov[X] = E[X - E[X]]^2 = E[X^2] - E[X]^2 \tag{2.12}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.3 State Space

The environment of a robot or system is characterized by *state* $\mathbf{x}$ defined as a vector of random variables. These state variables are *static* and non-changing or *dynamic* and changing over time t. The robot's pose and velocity are examples of dynamic state variables. We denote the state at time t as $\mathbf{x}_t$

A *complete* state would include all aspects of the environment and the robot itself, which is practically impossible. Therefore, only a subset of state variables are used for most system designs.

A probability distribution often characterizes the current state

$$p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \tag{2.13}$$

where $\mathbf{z}_t$ denotes **measurement data** at time t and $\mathbf{u}_t$ denotes **control data** at time t. The distinction between measurement and control data is important because they have different effects on the state: measurements increase our knowledge, while control (actions, e.g., in the form of movements) introduces noise and thus decreases our knowledge of the state.

If we assume the state $\mathbf{x}_{t-1}$ is complete, (2.13) can be simplified:

$$p(\mathbf{x}_t|\mathbf{x}_{0:t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{2.14}$$

Since it can be assumed that $\mathbf{x}_{t-1}$ already contains all previous controls and measurement knowledge. Likewise, if we assume $\mathbf{x}_t$ is complete, the following simplification holds for the same reason:

$$p(z_t|\mathbf{x}_{0:t}, \mathbf{z}_{1:t}, u_{1:t}) = p(\mathbf{z}_t|\mathbf{x}_t) \tag{2.15}$$

Since the state cannot often be measured directly, we must distinguish the *belief* of a state (what we infer from measurements) from the true state. The belief over a state $\mathbf{x}_t$ is denoted as

$$bel(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}). \tag{2.16}$$

Additionally, it is helpful to look at the belief state before incorporating $\mathbf{z}_t$ but after executing the control $\mathbf{u}_t$. This is dented as

$$\overline{bel}(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}). \tag{2.17}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

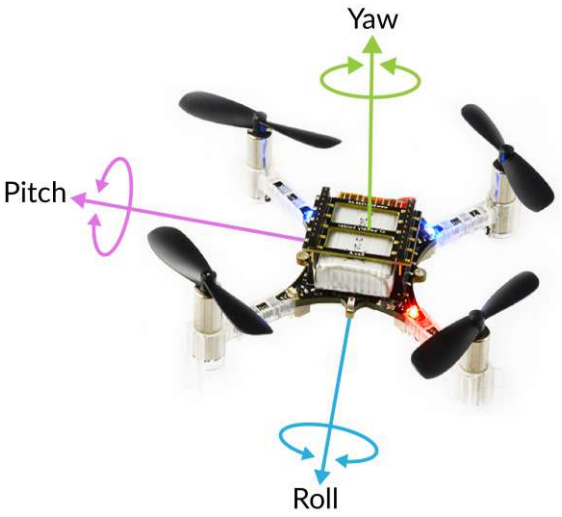


Figure 2.1: Image showing the orientation of a UAV in Euler angles, taken from [8].

2.4 Pose

2.4.1 Position

The *position* of an object describes its physical location in a given space. Two- and three-dimensional Euclidean spaces are the most common, depending on the intended use case. Typically, the position is defined as coordinates relative to a reference coordinate system.

Let $\mathbf{S}$ denote the three-dimensional space

$$\mathbf{S} = \{(x\ y\ z)^\top : x, y, z \in \mathbb{R}\}. \tag{2.18}$$

Using the *Cartesian coordinate system* the position P in the three-dimensional space $\mathbf{S}$ is given by

$$\mathbf{P} = (x\ y\ z)^\top, P \in \mathbf{S} \tag{2.19}$$

2.4.2 Orientation

The *orientation* of an object describes *how* it is placed in space. Rotational angles along the space axis can characterize it. In three-dimensional spaces, these angles are often referred to as *roll* θ, *pitch* ϕ and *yaw* ψ and are called *Euler angles*.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Alternatively, the orientation in three-dimensional spaces can also be defined using a *quaternion* **Q**

$$\mathbf{Q} = q_w + q_x i + q_y j + q_z k, \;\; q_x, q_y, q_z \in \mathbb{R} \tag{2.20}$$

where i, j and k are the fundamental quaternion units.

If the quaternion is normalized, i.e. $||q|| = \sqrt{q_w^2 + q_x^2 + q_y^2 + q_z^2} = 1$, quaternions are related to Euler angles by:

$$\begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix} = \begin{bmatrix} \text{atan2}(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)) \\ -\pi/2 + 2\text{atan2}(\sqrt{1 + 2(q_w q_y - q_x q_z)}, \sqrt{1 - 2(q_w q_y - q_x q_z)}) \\ \text{atan2}(2(q_w q_z + q_x q_z), 1 - 2(q_y^2 + q_z^2)) \end{bmatrix} \tag{2.21}$$

with

$$\text{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0, \\ \frac{\pi}{2} - \arctan(\frac{x}{y}) & \text{if } y > 0, \\ -\frac{\pi}{2} - \arctan(\frac{x}{y}) & \text{if } y < 0, \\ \arctan(\frac{y}{x}) \pm \pi & \text{if } x < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases} \tag{2.22}$$

2.4.3 Pose

The *pose* unites both position and orientation. As described in Section 2.3, the robots pose are dynamic state variables frequently part of the state space. This speaks for the importance of obtaining reliable position and orientation data.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.5 Localization

Robot localization describes the problem of determining a robot's pose in its environment. The uncertainties of the sensors, different environments, and the robot's movement are significant challenges to overcome. Generally, two types of localization can be distinguished: Absolute localization (also called global localization) and relative localization.

- **Absolute Localization**

  Absolute localization refers to positioning in a global or fixed coordinate system. The most prominent example is the global positioning system (GPS), which provides coordinates in a universal context and belongs to the category of global navigation satellite systems (GNSS).

- **Relative Localization**

  With relative localization, the position of an object is determined in relation to another object or several other objects. Relative localization does not provide global coordinates; it only allows for determining relative displacement vectors between objects.

In the remaining part of this Section, we present three localization systems that provide information about the pose or the position of an object in space. Those three were selected because they are relevant for this thesis.

2.5.1 ArUco fiducial markers

ArUco fiducial markers were first introduced in [19] as a camera pose estimation system for augmented reality and robot localization by Garrido-Jurado et al. The markers are detected using a camera, and their pose is estimated relative to the camera's pose.

ArUco Markers

ArUco markers are square graphics that are supposed to be recognized by a camera. They consist of a black outer border and an inner area representing a binary pattern using black and white squares (See Figure 2.2 for an example). The pattern is unique and orientation-independent for each marker, making it identifiable by its ID. The marker dictionary describes all possible configurations and is parameterized by the number of bits n encoded in each marker and the hamming distance d, specifying the minimum number of bits that differ between any two markers. For example, the smallest dictionary consists of 50 distinct markers that can be encoded by $n = 4 \times 4$ bits with a hamming distance of $d = 4$ or by $n = 5 \times 5$ bits with a hamming distance of $d = 8$ respectively.

The choice for d and n depends on the use case. Usually, smaller n are preferred if the goal is to detect the markers, as a lower resolution is required for recognition. If the correct distinction of different markers is more important than the recognition itself, higher d should be favored.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

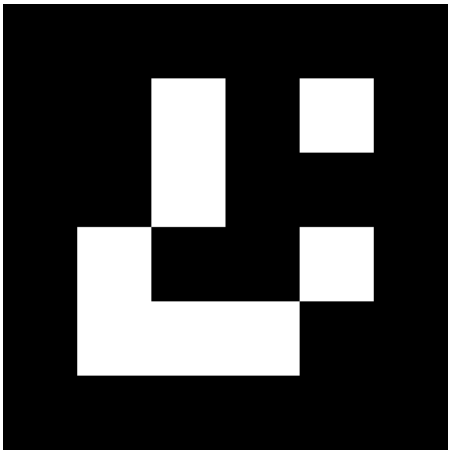


Figure 2.2: The ArUco marker with ID 4 from the dictionary with $n = 4 \times 4$ and $d = 4$, generated using [27]

Marker Detection

The marker detection is described in [19] using the following steps:

- **Image segmentation**: A local adaptive thresholding approach is applied to obtain all image borders (See Figure 2.3 (b)).
- **Contour extraction**: Using Suzuki and Abe's algorithm [50], contours are extracted from the thresholded image (See Figure 2.3 (c)).
- **Contour filtering**: By performing a polygonal approximation using the Douglas-Peucker algorithm [15], all contours that are not approximated to 4-vertex polygons are discarded (See Figure 2.3 (d)).
- **Removal of the perspective projection**: The perspective projection is removed for each marker respectively by computing the homography matrix, resulting in a frontal view of the rectangle area (See Figure 2.3 (e)).
- **Marker Code extraction**: Each marker is thresholded using Otsu's method [41] and divided into a grid according to the specified marker dictionary. Each grid element is assigned the value zero or one depending on the average pixel value inside of it. If the border bits are all zero (e.g., the border is black), all four rotations of the inner region are cross-checked with the dictionary (See Figure 2.3 (f)).
- **Error correction**: If the rotation-invariant Hamming distance between the markers is d, an error of at most $\lfloor (d-1)/2 \rfloor$ can be detected and corrected. This is done by calculating the distance to each marker in the dictionary and assigning the marker with the smallest distance, smaller or equal to $\lfloor (d-1)/2 \rfloor$ as the correct one.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

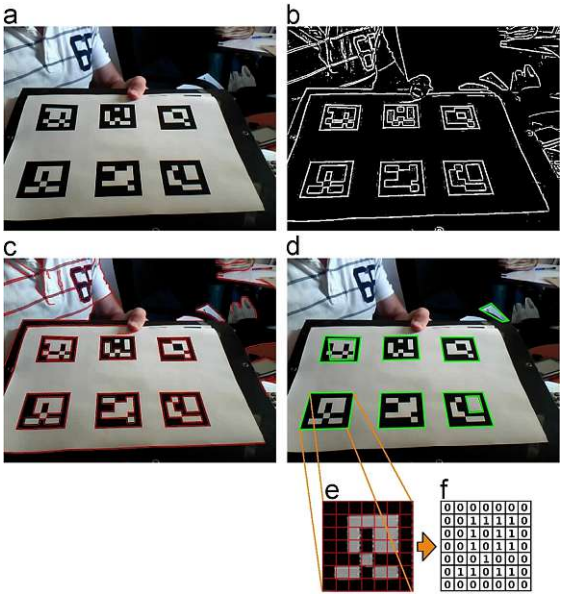


Figure 2.3: Image process for automatic marker detection, taken from Fig. 5 in [19]. (a) Original image. (b) Result of applying local thresholding. (c) Contour detection. (d) Polygonal approximation and removal of irrelevant contours. (e) Example of marker after perspective transformation. (f) Bit assignment for each cell.

Pose Estimation

The pose of a detected marker in relation to the camera (i.e., its 3D position in space) can be calculated using planar pose estimators. More specifically, this is done by iteratively minimizing the projection error of the four corners of the markers using the Levenberg-Marquardt algorithm [36]. The corners of the markers are estimated by performing a linear regression with its side pixels and calculating the resulting intersections. To perform the pose estimation, the camera matrix (i.e., its intrinsic parameters) and the distortion coefficients of the camera lens have to be known.

Pose estimation using ArUco markers is widely supported in libraries such as OpenCV [11] and needs relatively low computational cost. Pairing that with its affordability (requiring only an off-the-shelf camera and printed markers) and high accuracy in detecting the pose in two dimensions has made ArUco markers a popular choice for pose estimation. However, there are also disadvantages: The detection requires the markers to be placed on objects without being partially or fully occluded. Additionally, the effective range for detection is limited, depending on the resolution and quality of the camera. The depth information of a recognized pose is also much less accurate than the position on the x/y plane.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Figure 2.4: Image showing four passive markers with reflective material.

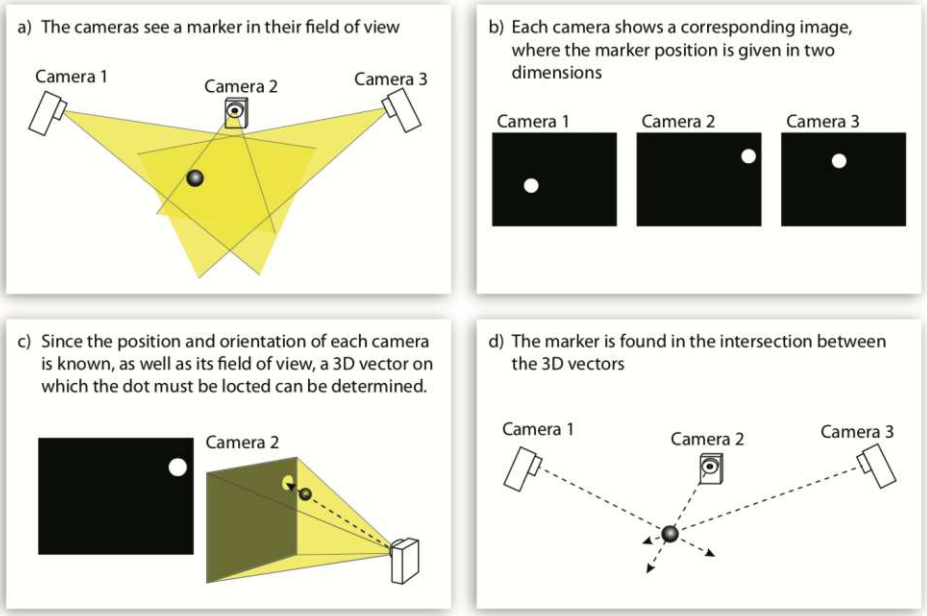


Figure 2.5: Image showing how motion capture works, taken from [18].

2.5.2 Optical Motion Tracking using spherical markers

Optical Motion Tracking is a technology that uses cameras to capture the movement of objects in physical space. Marker-based systems use small reflective spheres attached to the object of interest. Those spheres are also called markers but are unrelated to the previously mentioned ArUco markers. This marker can be active and emit infrared light or passive and is coated with a reflective material (See Figure 2.4). Infra-red cameras are used to record a two-dimensional image. The infrared cameras must be equipped with infrared light sources if passive spheres are used.

The markers can be identified by thresholding the image if they are in the camera's field of view. If the position and orientation of the camera are known, a 3D vector on which the marker must be located can be estimated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

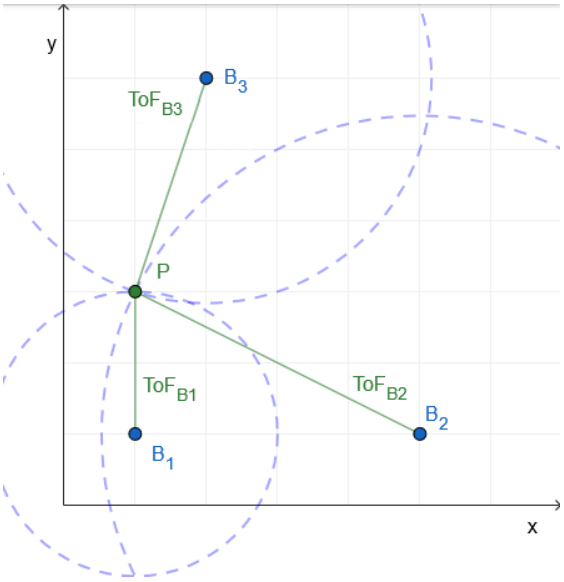


Figure 2.6: Example showing the visualization of ToF measurements (in green) within a two-dimensional plane. Each circle represents the possible position of a single ToF measurement.

The three-dimensional position can be identified by the intersection of 3D vectors of the same marker from multiple cameras. See Figure 2.5 for a visual representation.

A group of markers can now be used to define a single object. If the markers in this group are arranged asymmetrically, each configuration becomes identifiable so that the pose can be derived.

2.5.3 Time of Arrival (ToA)

Time of Arrival (ToA) is a ranging method for determining the distance between two devices that can communicate with each other. It calculates the Time of Flight (ToF) using the time a signal was sent from the first device and the time a signal was received by the second device. Since the signal propagation velocity is a known constant, the ToF can directly be converted to distances.

The ToF measurement provides a direct estimation of the distance between two devices. For a device to estimate its position within a two-dimensional plane, it must obtain ToF measurements of at least three devices with predetermined fixed positions. Each of these measurements can be visualized as a circle around this known position, with the ToF measurement corresponding to the circle's radius. At least three of these circles are needed to intersect at a single point, representing the estimated position of the unknown device (See Figure 2.6).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

To estimate the position of an unknown device in three-dimensional space, it is necessary to know the ToF to at least four devices. One way to represent this is by using spheres instead of circles. The ToA method is subject to uncertainties. Maintaining precise alignment between the clocks of different devices over time is difficult but necessary for determining the ToF. Additionally, temperature fluctuations can also influence the calculations, as they can affect the propagation speed of signals.

Two Way Ranging Protocol (TWR)

The two-way ranging protocol by Bitcraze [10] implements ToA for positioning:

Four messages are exchanged for the ToF calculation; two are sent from device A and two from device B. In addition, both devices need to record the transmission and reception times.

1. POLL message: Send from device A to device B. Device A records the transmission time T1 of the first message, and device B records the reception time R1.
2. ANSWER message: Send from device B to device A as a response to the POLL message. Device B records the transmission time T2, and device A records the reception time R2.
3. FINAL message: Send from device A to device B as a response to the ANSWER message: Device A records the transmission time T3, and device B records the reception time R3.
4. REPORT message: Send from device B to device A. It contains the recorded timestamps of device B (R1, T2, and R3) and is the last message of the exchange.

Using all the recorded timestamps, device A can calculate its reply-time T_{replyA} and the reply-time of device B T_{replyB} as well as both round-trip times $T_{roundABA}$ and $T_{roundBAB}$:

$$T_{replyB} = T2 - R1 \tag{2.23}$$
$$T_{replyA} = T3 - R2 \tag{2.24}$$
$$T_{roundABA} = R2 - T1 \tag{2.25}$$
$$T_{roundBAB} = R3 - T2 \tag{2.26}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

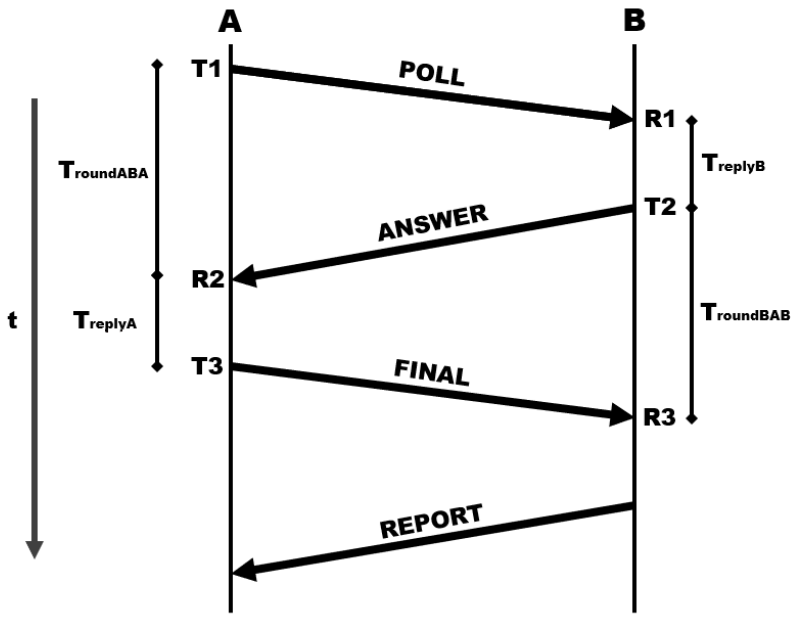


Figure 2.7: The two-war ranging protocol. Devices A and B both exchange two messages and record their respective transmission and reception times, which are then used to estimate the ToF.

The ToF can be estimated using the following formula:

$$ToF = \frac{T_{roundABA} \cdot T_{roundBAB} - T_{replyA} \cdot T_{replyB}}{T_{roundABA} + T_{roundBAB} + T_{replyA} + T_{replyB}} \tag{2.27}$$

With the TWR protocol, a device must actively communicate with at least four other devices at known positions to determine its position. This protocol is ideal for environments where only a single device wants to determine its position. However, it does not scale well if multiple devices are introduced, as they would all need to communicate, leading to saturation of the available communication bandwidth and potential signal interference.

2.5.4 Time Difference of Arrival (TDoA)

Time Difference of Arrival (TDoA) describes the time difference between the Time of Arrival (ToA) of a signal recorded by multiple receivers.

A device C can estimate TDoA by passively listening to the communication of the two devices A and B doing TWR and recording the reception times of the POLL, ANSWER, and FINAL message as well as the recorded timestamps and calculated ToF_{AB} from device A.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

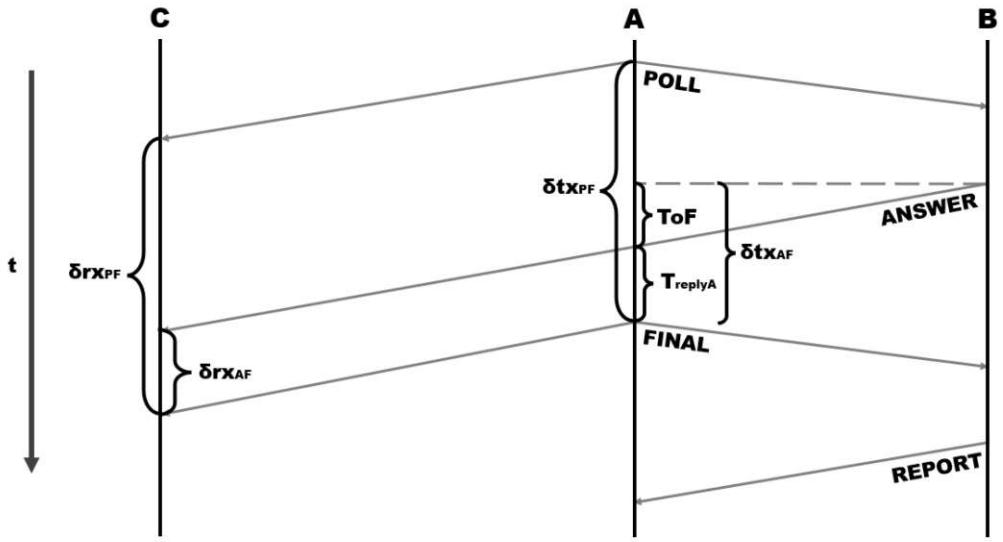


Figure 2.8: The TDoA protocol. Devices A and B perform TWR, letting Device C passively listen to their message exchange. By comparing its times of arrival with the ones derived from the message contents, the TDoA is estimated.

Since physical devices do not have a perfect clock, the drift between the clocks of device A and device B has to be compensated by a compensation factor $\alpha = \delta rx_{PF}/\delta tx_{PF}$ where δrx_{PF} is the difference in time between the reception of the POLL message and the reception of the FINAL message in device C and δtx_{PF} is the difference in time between the transmission of those messages in device A. The clock drift of device B does not need to be considered since only timestamps from device A are necessary for the calculation.

Let δrx_{AF} be the difference in time between the reception of the ANSWER message and the reception of the FINAL message in the clock of device C and $\delta tx_{AF} = ToF_{AB} + T_{replyA}$ be the difference in time between the transmission of those messages in device A in the clock of device C.

The TDoA (i.e., the difference in time of flights) can be estimated using the following formula:

$$TDoA = \delta rx_{AF} - (\alpha \cdot \delta tx_{AF}) \tag{2.28}$$

Similar to the visualization of the ToF as a circle, TDoA can be visualized as a hyperbola between the two corresponding devices with known positions, where the TDoA value is constant. In a two-dimensional plane, three TDoA measurements are needed to estimate the position of the unknown device. In a three-dimensional space, the required number is four.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

	fiducial markers	*optical motion tracking*	*UWB systems*
accuracy	depends on the used camera, poor depth estimation	high accuracy (4 mm)	medium accuracy (10 cm)
costs	cheap (starting at 10€)	very expensive (2000€-4000€ per camera)	expensive (200€ per node)
pose estimation	yes	yes, if at least three marker are used	no
scope	relative localization	absolute localization	absolute localization
minimal setup	single camera	two cameras	four+one UWB radios

Table 2.1: Comparison of the different Localization systems, showing their differences in accuracy, costs, capability of pose estimation, scope and minimal setup requirements

The main improvement of TDoA over TWR is the scalability when multiple devices are used. TDoA allows devices to determine their positions passively, just by listening to the communication between the reference points. This reduces communication overhead and avoids signal collisions, making TDoA more suitable in those scenarios.

2.5.5 Comparison of the different Localization Systems

Table 2.1 serves as a summary of the different localization systems by directly comparing them in several key areas.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.6 Kalman Filter

R. E. Kalman developed the Kalman filter [28] in 1960, and it is one of the most important algorithms for state estimation in the category of Bayes filters. It uses a series of noisy measurements and the assumed system dynamics to estimate the belief state continuously. This method is suitable to use when the state variables cannot be measured directly, or different measurements must be fused. It is optimal assuming that the noise and the system dynamics are Gaussian.

The following Sections are a continuation of the summary of [51], introducing the Kalman filter and how it can be used to integrate sensor measurement into the belief state of a system.

Bayes Filter

The Bayes filter is a recursive algorithm to calculate the belief distribution from measurement and control data, starting from an initial belief $bel(\mathbf{x}_0)$ at time $t = 0$. It assumes the state is complete and the controls are time-independent.

Using the Bayes rule from Equation (2.9) we can expand Equation (2.16) to

$$bel(\mathbf{x}_t) = \eta \; p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \; p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \tag{2.29}$$

and further simplify it using Equation (2.15) and Equation (2.17):

$$bel(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t}, \mathbf{u}_{1:t}) = \eta \; p(\mathbf{z}_t|\mathbf{x}_t) \; \overline{bel}(\mathbf{x}_t) \tag{2.30}$$

Likewise we can use Equation (2.7) to expand Equation (2.17)

$$\overline{bel}(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \; p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) \; d\mathbf{x}_{t-1} \tag{2.31}$$

And further simplify it using Equation (2.14) and Equation (2.16):

$$\overline{bel}(\mathbf{x}_t) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, u_t) \; bel(\mathbf{x}_{t-1}) \; d\mathbf{x}_{t-1} \tag{2.32}$$

It has to be noted that the following holds because $\mathbf{u}_t$ does not perdate the state $\mathbf{x}_{t-1}$:

$$p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}) = p(\mathbf{x}_{t-1}|\mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}) \tag{2.33}$$

Using both Equation (2.30) and Equation (2.32), the algorithm of the Bayes filter can be defined as seen in Algorithm 2.1. The calculation of $\overline{bel}(\mathbf{x}_t)$ gets called the *prediction* step and the calculation of $bel(\mathbf{x}_t)$ using $\overline{bel}(\mathbf{x}_t)$ gets called the *correction* step.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The Bayes filter is generally impractical since the state can not be complete and, therefore, must be approximated. Furthermore, continuous state spaces would be computationally unfeasible.

Algorithm 2.1: Bayes Filter

input : $bel(\mathbf{x}_{t-1})$, $\mathbf{u}_t$, $\mathbf{z}_t$

output : $bel(\mathbf{x}_t)$

forall x_t **do**
  $\overline{bel}(\mathbf{x}_t) \longleftarrow \int p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\ bel(\mathbf{x}_{t-1})\ d\mathbf{x}_{t-1}$;
  $bel(\mathbf{x}_t) \longleftarrow \eta\ p(\mathbf{z}_t|\mathbf{x}_t)\ \overline{bel}(\mathbf{x}_t)$;
end
return $bel(x_t)$

Gaussian Filter

Gaussian Filters are part of the family of Bayes filters where beliefs are limited to multivariate *normal* (i.e., *Gaussian*) distributions. A multivariate normal distribution is defined as

$$p(\mathbf{x}) = \frac{1}{\sqrt{|2\pi\boldsymbol{\Sigma}|}}\ e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.34}$$

where the density $p(\mathbf{x})$ depends only on the mean $\boldsymbol{\mu}$ (with $\dim(\boldsymbol{\mu}) = \dim(\mathbf{x})$) and the covariance $\boldsymbol{\Sigma}$ (with $\dim(\boldsymbol{\Sigma}) = \dim(\mathbf{x}) \times dim(\mathbf{x})$). This reduces the complexity significantly, making it feasible to compute.

For Gaussian filters $bel(\mathbf{x}_t)$ is represented by mean $\boldsymbol{\mu}_{\mathbf{x}_t}$ with the covariance $\boldsymbol{\Sigma}_{\mathbf{x}_t}$ determining its uncertainty.

2.6.1 Kalman Filter

The Kalman filter implements the Gaussian filter for linear systems with continuous states. To guarantee that the calculated belief states are still Gaussian at every time step, the following has to be true:

- The *complete state assumption* must hold, meaning past and future data are independent if the current state $\mathbf{x}_t$ is known (if this can not be guaranteed, it should be ensured that the effect of all unmodeled state variables is close to random).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- The state transition probability $p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1})$ can be expressed as a *linear* function with Gaussian noise $\boldsymbol{\epsilon}_t$. This is given if the system dynamics are linear:

$$\mathbf{x}_t = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \boldsymbol{\epsilon}_t \tag{2.35}$$

  Let $n = dim(\mathbf{x}_t)$ and $m = dim(\mathbf{u}_t)$ than $\mathbf{A}_t$ is a square matrix of size $n \times n$ and $\mathbf{B}_t$ is a matrix of size $n \times m$. $\boldsymbol{\epsilon}_t$ is a vector of size n that models randomness in the state transition with zero mean and $\mathbf{R}_t$ as its covariance. The matrix $\mathbf{A}_t$ describes how the state evolves from time $t-1$ to t, ignoring controls or noise. The matrix $\mathbf{B}_t$ describes the effect of the control $\mathbf{u}_t$ to the state from time $t-1$ to t.

$$p(\mathbf{x}_t|\mathbf{u}_t, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t, \mathbf{R}_t) \tag{2.36}$$

- The measurement probability $p(\mathbf{z}_t|\mathbf{x}_t)$ can be expressed as a *linear* function with Gaussian noise $\boldsymbol{\delta}_t$:

$$\mathbf{z}_t = \mathbf{C}_t\mathbf{x}_t + \boldsymbol{\delta}_t \tag{2.37}$$

  Let $k = dim(\mathbf{z}_t)$ than $\mathbf{C}_t$ is a matrix of size $k \times n$. Like state transition randomness, $\boldsymbol{\delta}_t$ is a vector of size n that quantifies the measurement noise with zero means and $\mathbf{Q}_t$ as its covariance. The matrix $\mathbf{C}_t$ describes how a measurement $\mathbf{z}_t$ maps to the state at time t.

$$p(\mathbf{z}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{C}_t\mathbf{x}_t, \mathbf{Q}_t) \tag{2.38}$$

- The initial belief $bel(\mathbf{x}_0)$ is normal distributed with mean $\boldsymbol{\mu}_0$ and covariance $\boldsymbol{\Sigma}_0$:

$$bel(\mathbf{x}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{2.39}$$

The non-trivial proof of this is omitted. The interested reader is referred to [51], Section (3.2.4).

The Kalman filter algorithm gives us the belief $bel(\mathbf{x}_t)$ at time t represented by $\boldsymbol{\mu}_t$ and $\boldsymbol{\Sigma}_t$ using the previous belief $bel(\mathbf{x}_{t-1})$ at time $t-1$ represented by $\boldsymbol{\mu}_{t-1}$ and $\boldsymbol{\Sigma}_{t-1}$ as well as the control $\mathbf{u}_t$ and the measurement $\mathbf{z}_t$ at time t as inputs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 2.2: Kalman Filter

input : $\boldsymbol{\mu}_{t-1}$, $\boldsymbol{\Sigma}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$

output : $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

1 $\overline{\boldsymbol{\mu}}_t \longleftarrow \mathbf{A}_t\boldsymbol{\mu}_{t-1} + \mathbf{B}_t\mathbf{u}_t$;

2 $\overline{\boldsymbol{\Sigma}}_t \longleftarrow \mathbf{A}_t\boldsymbol{\Sigma}_{t-1}\mathbf{A}_t^T + \mathbf{R}_t$;

3 $\mathbf{K}_t \longleftarrow \overline{\boldsymbol{\Sigma}}_t\mathbf{C}_t^T(\mathbf{C}_t\overline{\boldsymbol{\Sigma}}_t\mathbf{C}_t^T + \mathbf{Q}_t)^{-1}$;

4 $\boldsymbol{\mu}_t \longleftarrow \overline{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}_t\overline{\boldsymbol{\mu}}_t)$;

5 $\boldsymbol{\Sigma}_t \longleftarrow (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\overline{\boldsymbol{\Sigma}}_t$;

6 **return** $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

Line 1 and line 2 of Algorithm 2.2 implement the prediction step by calculating $\overline{bel}(\mathbf{x}_t)$ based on (2.36): Line 1 is derived by substituting $\mathbf{x}_{t-1}$ with $\boldsymbol{\mu}_{t-1}$. In contrast, Line 2 predicts the error covariance $\overline{\boldsymbol{\Sigma}}_t$ using the system dynamics in matrix $\mathbf{A}_t$ as well as the noise covariance matrix $\mathbf{R}_t$ that accounts for the randomness in the state transition.

Lines 3, 4, and 5 implement the update step that calculates $bel(\mathbf{x}_t)$ using $\overline{bel}(\mathbf{x}_t)$ and the measurement $\mathbf{z}_t$. First, the Kalman gain $\mathbf{K}_t$ is calculated in Line 3. It determines the effect the measurement $\mathbf{z}_t$ should have on the predicted state $\overline{bel}(\mathbf{x}_t)$. If $\mathbf{Q}_t$ is high, the Kalman gain will be small, resulting in a small influence on the measurement. In contrast, if $\overline{\boldsymbol{\Sigma}}_t$ is high, the prediction has to be assumed to be unreliable, leading to a more considerable Kalman gain that allows the measurement to have a more significant influence. After that, it is used in Line 4 to adjust the predicted $\overline{\boldsymbol{\mu}}_t$ by the difference between the actual measurement $\mathbf{z}_t$ and the predicted measurement $\mathbf{C}_t\overline{\boldsymbol{\mu}}_t$. Line 5 updates the predicted error covariance $\overline{\boldsymbol{\Sigma}}_t$ based on the information gained from the measurement.

2.6.2 Extended Kalman Filter

In robotics, the assumption that state transitions and measurements are linear is often not feasible. The extended Kalman filter (EKF) tries to overcome this restriction by only approximating the belief $bel(\mathbf{x}_t)$. This allows the next state probability and the measurement probability to be defined by the nonlinear functions g resp. h:

$$\mathbf{x}_t = g(\mathbf{u}_t, \mathbf{x}_{t-1}) + \boldsymbol{\epsilon}_t \tag{2.40}$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \boldsymbol{\delta}_t \tag{2.41}$$

The key idea is to use *linearization* using first order *Taylor expansion* around a point a:

$$f(x) \approx f(a) + f'(a)(x - a) \tag{2.42}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

EKFs use the current most likely state for linearization, which is parameterized by $\boldsymbol{\mu}_{t-1}$ and $\mathbf{u}_t$. For function $g(\mathbf{u}_t, \mathbf{x}_{t-1})$ this gives us

$$g(\mathbf{u}_t, \mathbf{x}_{t-1}) \approx g(\mathbf{u}_t, \boldsymbol{\mu}_{t-1}) + \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \tag{2.43}$$

with

$$\mathbf{G}_t = \frac{\partial g(\mathbf{u}_t, \boldsymbol{\mu}_{t-1})}{\partial \boldsymbol{\mu}_{t-1}}. \tag{2.44}$$

Let $n = dim(\mathbf{x}_t)$ than $\mathbf{G}_t$ is a square matrix of size $n \times n$. This matrix is called the *Jacobian* of the function g.

Similarly also function $h(\mathbf{x}_t)$ gets linearized around $\overline{\boldsymbol{\mu}}_t$:

$$h(\mathbf{x}_t) \approx h(\overline{\boldsymbol{\mu}}_t) + \mathbf{H}_t(\mathbf{x}_t - \overline{\boldsymbol{\mu}}_t) \tag{2.45}$$

with

$$\mathbf{H}_t = \frac{\partial h(\mathbf{x}_t)}{\partial \mathbf{x}_t}. \tag{2.46}$$

Using both (2.43) and (2.45), the algorithm of the Kalman filter can be extended to the algorithm of the EKF (Algorithm 2.3).

Algorithm 2.3: Extended Kalman Filter

input : $\boldsymbol{\mu}_{t-1}$, $\boldsymbol{\Sigma}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$

output : $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

1 $\overline{\boldsymbol{\mu}}_t \longleftarrow (\mathbf{u}_t, \mathbf{x}_{t-1})$;

2 $\overline{\boldsymbol{\Sigma}}_t \longleftarrow \mathbf{G}_t\boldsymbol{\Sigma}_{t-1}\mathbf{G}_t^T + \mathbf{R}_t$;

3 $\mathbf{K}_t \longleftarrow \overline{\boldsymbol{\Sigma}}_t\mathbf{H}_t^T(\mathbf{H}_t\overline{\boldsymbol{\Sigma}}_t\mathbf{H}_t^T + \mathbf{Q}_t)^{-1}$;

4 $\boldsymbol{\mu}_t \longleftarrow \overline{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - h(\overline{\boldsymbol{\mu}}_t))$;

5 $\boldsymbol{\Sigma}_t \longleftarrow (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\overline{\boldsymbol{\Sigma}}_t$;

6 **return** $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

2.6.3 Unscented Kalman Filter

The *Unscented* Kalman Filter (UKF) from Julier et al. [26] uses a different approach compared to the EKF to handle nonlinear systems: Instead of approximating (e.g., linearizing) the nonlinear function, the Gaussian distribution gets approximated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Unscented Transform

The *unscented Transform* uses points, called *sigma points*, to propagate a random variable $\mathbf{x}$ through a non-linear function $f(\mathbf{x})$. This is done by the following steps:

1. The n-dimensional random variable $\mathbf{x}$ with mean $\boldsymbol{\mu}_\mathbf{x}$ and covariance $\boldsymbol{\Sigma}_\mathbf{x}$ gets approximated with by $2n+1$ sigma points:

$$\begin{aligned} \mathcal{X}_0 &= \mu & W_0 &= \kappa/(n+\kappa) \\ \mathcal{X}_i &= \mu + (\sqrt{(n+\kappa)\Sigma})_i & W_i &= 1/2(n+\kappa) \\ \mathcal{X}_{i+n} &= \mu - (\sqrt{(n+\kappa)\Sigma})_i & W_{i+n} &= 1/2(n+\kappa) \end{aligned} \tag{2.47}$$

   where $\kappa \in \mathbb{R}$, $\mathcal{X}_i$ is the ith sigma point and W_i is the weight of the ith sigma point.

2. Each sigma point gets transformed individually by the function f:

$$\mathcal{Y}_i = f(\mathcal{X}_i) \tag{2.48}$$

3. The transformed mean μ_y and covariance Σ_y are calculated from the transformed sigma points:

$$\mu_y = \sum_{i=0}^{2n} W_i \mathcal{Y}_i \tag{2.49}$$

$$\Sigma_y = \sum_{i=0}^{2n} W_i \{\mathcal{Y}_i - \mu_y\}\{\mathcal{Y}_i - \mu_y\}^T \tag{2.50}$$

The paper shows that the means are calculated to a higher order of accuracy compared to EKF, while the covariance is calculated to the same order of accuracy. Additionally, higher-order information is partially incorporated, which also results in an improvement.

The Filter Algorithm

For the unscented Kalman filter, the state vector $\mathbf{x}_t$ gets augmented with the process and measurement noise $\mathbf{v}_t$:

$$\mathbf{x}_t^\alpha = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{v}_t \end{bmatrix} \tag{2.51}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The next state probability and the measurement probability must be adapted to the extended state vector

$$\mathbf{x}_t^\alpha = g(\mathbf{u}_t, \mathbf{x}_{t-1}^\alpha) \tag{2.52}$$

$$\mathbf{z}_t = h(\mathbf{x}_t^\alpha), \tag{2.53}$$

and the corresponding sigma points are defined using Equation (2.47).

By following the principles of the Kalman filter and applying the unscented transform to sigma points, the algorithm of the Unscented Kalman filter can be formed:

Algorithm 2.4: Unscented Kalman Filter

input : $\mathcal{X}_{t-1}^\alpha$, $\mathbf{u}_t$, $\mathbf{z}_t$

output: $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

$\mathcal{X}_t^\alpha \longleftarrow g(\mathbf{u}_t, \mathcal{X}_{t-1}^\alpha)$;

$\overline{\boldsymbol{\mu}}_t \longleftarrow \sum_{i=0}^{2n^\alpha} W_i \mathcal{X}_{i,t}^\alpha$;

$\overline{\boldsymbol{\Sigma}}_t \longleftarrow \sum_{i=0}^{2n^\alpha} W_i \{\mathcal{X}_{i,t} - \overline{\boldsymbol{\mu}}_t\}\{\mathcal{X}_{i,t} - \overline{\boldsymbol{\mu}}_t\}^T$;

$\mathcal{Z}_t^\alpha \longleftarrow h(\mathcal{X}_t^\alpha)$;

$\overline{\mathbf{z}}_t \longleftarrow \sum_{i=0}^{2n^\alpha} W_i \mathcal{Z}_{i,t}^\alpha$;

$\overline{\boldsymbol{\Sigma}}_t^{z,z} \longleftarrow \sum_{i=0}^{2n^\alpha} W_i \{\mathcal{Z}_{i,t} - \overline{\mathbf{z}}_t\}\{\mathcal{Z}_{i,t} - \overline{\mathbf{z}}_t\}^T$;

$\overline{\boldsymbol{\Sigma}}_t^{x,z} \longleftarrow \sum_{i=0}^{2n^\alpha} W_i \{\mathcal{X}_{i,t} - \overline{\boldsymbol{\mu}}_t\}\{\mathcal{Z}_{i,t} - \overline{\mathbf{z}}_t\}^T$;

$\mathbf{K}_t \longleftarrow \overline{\boldsymbol{\Sigma}}_t^{x,z} \overline{\boldsymbol{\Sigma}}_t^{z,z\,-1}$;

$\boldsymbol{\mu}_t \longleftarrow \overline{\boldsymbol{\mu}}_t + \mathbf{K}_t(\mathbf{z}_t - \overline{\mathbf{z}}_t\})$;

$\boldsymbol{\Sigma}_t \longleftarrow \overline{\boldsymbol{\Sigma}}_t - \mathbf{K}_t \overline{\boldsymbol{\Sigma}}_t^{z,z} \mathbf{K}_t^T$;

return $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

CHAPTER 3

System design

As introduced in Section 1.2, this work is pursuing two objectives:

1. Improve the localization of the UAV with respect to the ground vehicle.
2. Design a high-level landing controller to land the UAV on a moving ground vehicle.

Achieving the first goal is a necessity for the second goal.

In the following sections, we will present the components and hardware involved and our approach to achieving the objectives.

3.1 Robotic Platform

The robotic platform refers to the base framework of our system design and includes the hardware, software, and tools used for development and implementation.

3.1.1 Robot Operating System (ROS)

The robot operating System (ROS) is a software framework that provides a structured communication layer above the operating systems of interconnected computing devices. In their paper [44], Quigley et al. describe the philosophical goals of ROS 1 as follows:

- **Peer-to-Peer**: A ROS system consists of a set of processes, called nodes, that can run on multiple different hosts. The nodes are connected using a peer-to-peer network that allows them to communicate with one another. A name service called the ROS master provides a lookup mechanism that enables the ROS nodes to find each other.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The sender of messages, called publishers, send their message to the whole network using a *topic*, categorizing the message. The receivers, called subscribers, are unknown by the publishers. By subscribing to a topic, they decide to receive the corresponding messages.

- **Multi-lingual**: ROS is designed to be language-neutral and supports four different programming languages. By implementing ROS natively in each language and defining messages language-neutral using a simple interface definition language, each node can be implemented independently with guaranteed interoperability.
- **Tool-based**: ROS uses a microkernel architecture where different tools are used to build and run all ROS components. This helps reduce the complexity because the nodes themselves can be designed to do one simple task only.
- **Thin**: The ROS core is designed to be as light as possible, requiring all middleware, drivers, and algorithm development to occur in standalone libraries with no ROS dependencies. ROS only acts as a common interface for configuration and data routing.
- **Free and Open-Source**: ROS is free and publicly available under the terms of the BSD license. This allows ROS to be used commercially and non-commercially.

The disadvantages of ROS are the single point of failure of the ROS master, missing built-in security mechanisms, and inconsistent message deliveries over lossy links. Because of that, ROS 2 was redesigned from the ground up using an open standard for communications and support for real-time messaging and state-of-the-art message encryption [34]. With the use of *Peer-to-Peer* discovery eliminated the need of the central name server.

In this work, ROS 1 and ROS 2 were used as the software framework of choice: We use ROS 1 in the simulation since the UAV simulator CrazyS [47] was not supported in ROS 2. The main disadvantages of ROS 1 primarily affect the physical world, and because of that, this choice has no noticeable downsides.

For our real-world experiment, ROS 2 runs on the computing platform of the ground vehicle and controls both the ground vehicle and the UAV.

3.1.2 F1Tenth Race Car

The F1TENTH race car[40] is an open-source 1/10 scale autonomous vehicle testbed that acts as the ground vehicle for this work. The vehicle is a modified RC-car based on the *Traxxas Slash 4x4 Premium Chassis* [52] that is equipped with an embedded computing platform (NVIDIA Jetson). Due to the openness of the design, it can be expanded with various sensors as required. Typical a *Hokuyo UST-10LX LIDAR* is included. The car uses a high torque brushless DC motor and a Lithium Polymer (LiPo) battery pack that is controlled by an electronic speed controller (*VESC* [53]) that also includes an inertial measurement unit (*IMU*).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

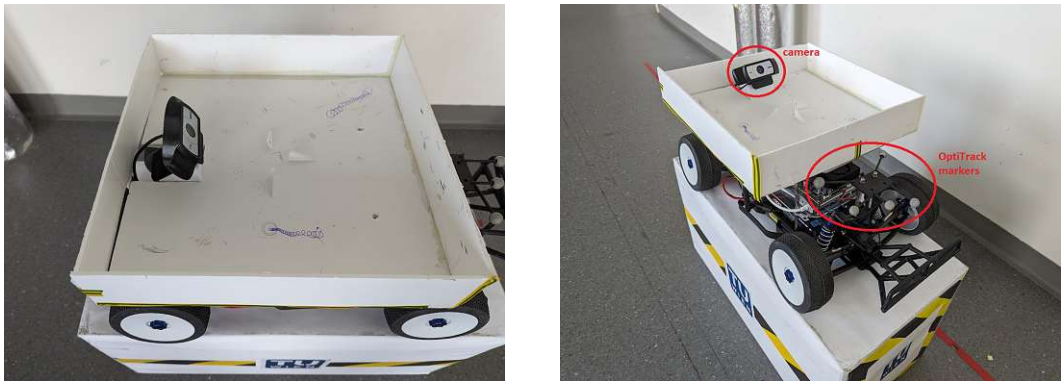


Figure 3.1: F1TENTH race care with a landing platform, OptiTrack markers, and camera.

With the help of a motion-capturing system (See Section 2.5.2), the vehicle's position can be accurately determined.

For this work, OptiTrack (Prime 13 cameras with MotiveTracker 1.10.1) is used as the motion-capturing system to determine the position of the ground vehicle. Furtado et al. [17] showed that the average error of this system is 4 mm.

The car's standard configuration is extended to include a camera (Logitech C930e) with a maximum resolution of 1920x1080 at 30 frames per second. The camera as well as a *CrazyRadio*, a long-range open USB radio dongle based on the *nRF24LU1+* from *Nordic Semiconductor* (used to control the UAV) are connected to the car via a USB hub. Control commands are sent from the human operator using the *Logitech F710* wireless gamepad.

F1Tenth System

The *ROS f1tenth_system* [14] package contains the drivers for the f1tenth race car. The car itself can be controlled using a Joystick. The button presses are published at the ROS topic */joy* and converted to velocity and drive angle values published to the */ackermann_cmd* topic.

3.1.3 Crazyflie

The UAV used in this work is a *Crazyflie 2.1* [21], which, from now on, we will just be referred to as Crazyflie. It is a versatile open-source flight development platform that is very lightweight (27g) and equipped with a low-latency 2.4GHz radio capable of a line-of-sight range of up to 1 km. The UAV's average flight time is 7 minutes, and it can carry a payload of up to 15g. It has the following dimension, measured motor-to-motor: $92 \times 92 \times 29\ mm$ ($W \times H \times D$).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

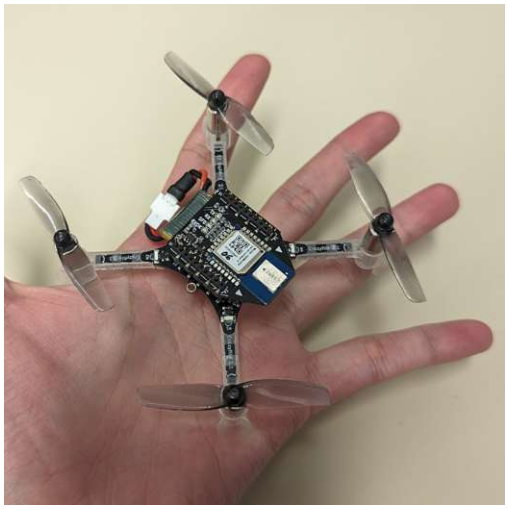


Figure 3.2: The Crazyflie UAV with the hardware upgrade on a human hand.

All processing is split between two microcontrollers (MCUs). The main application is handled by an *STM32F405 Cortex-M4* MCU, running with 168MHz, 192kb SRAM, and 1Mb flash. An *nRF51822 Cortex-M0* MCU handles power management and radio communication. It runs with 32Mhz, 16kb SRAM, and 128kb flash. A LiPo battery provides power and can be charged using the micro-USB connector via the on-board charger. The USB connector also doubles as the UAV programming interface.

The Crazyflie features an (IMU) with a three-axis accelerometer, gyro, magnetometer, and a high-precision pressure sensor. Additionally, it exposes various MCU pins over its extension connector, capable of I2C, SPI, and UART communication. It makes it possible to attach extension boards that add functionality, such as an ultra-wideband (UWB) radio module.

The firmware running on the Crazyflie is based on FreeRTOS [5], a real-time operating system (RTOS) kernel for microcontrollers. It enables multitasking using a scheduler and inter-task communication. Utilizing a deterministic scheduler turns the operating system into a real-time operating system, as time requirements can be guaranteed.

Figure 3.3 shows a simplified view of Crazyflie's operation.

We track the position of the Crazyflie using the *Loco Positioning System*, which is based on UWB radios (DW1000) that can measure the distance between each other. The radio modules are placed on the UAV and predefined coordinates. Using the distance measurements and the locations, the position of the UAV is calculated using TDoA (See Section 2.5.4). We use TDoA mode instead of TWR mode because it does not necessitate two-way communication between the UAV and the anchors. The main benefit of this is the possibility of supporting any number of UAVs.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

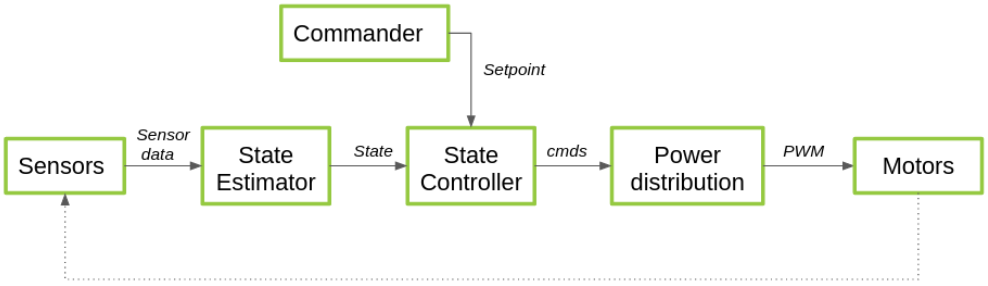


Figure 3.3: A high-level overview of the Crazyflie firmware, taken from [9].

Hardware Update

The base Crazyflie UAV is upgraded using bigger motors that provide an additional 20 grams of thrust. The upgrade allows for a larger payload but at the expense of roughly 15% less flight time (See Figure 3.2).

Extended Kalman Filter of the Crazyflie

The firmware of the Crazyflie implements an EKF (See Section 2.6.2) that is based on the work of Mueller et al. [37], [38].

The state vector $\mathbf{x}$ contains the following nine variables:

- x, y, z: the quadrocopter's position in the global frame
- ω_x, ω_y, ω_z: the quadrocopter's velocity in its body frame
- δ_0, δ_1, δ_2: the attitude error that encodes the uncertainty about its orientation.

The measurements of three different sensors are used in the update step:

- Angular Rate Gyroscope
- Accelerometer
- UWB Range Measurements

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Crazyswarm

Crazyrwarm [43] is a system architecture that allows the indoor control of multiple Crazyflie quadcopters in tight synchronized formations. It provides several advantages compared to Bitcraze's native *cflib* [7], the official Crazyflie python library, like motion capture integration, broadcast messages, and a ROS interface. The ROS interface is the determining factor why Crazyswarm was chosen for this work since the only other alternative *crazyflie_ros* [23] is deprecated and no longer maintained.

Our work uses the currently newest iteration *crazyswarm2* with the C++ backend that uses ROS2.

The core of the Crazyswarm2 architecture is the Crazyflie server. It acts as a ROS node and connects to the Crazyflie quadcopter via a Crazyradio. It provided various interfaces for communication and control of the Crazyflie. The most important ones, which are also used in this work, are the following:

- Takeoff (/cf1/takeoff): By providing height and duration, a takeoff command is sent to the UAV to take off and fly to the provided height in the specified amount of time.
- Land (/cf1/land): By providing height and duration, a land command is sent to the UAV to power down the motors and stop them at the provided height in the specified amount of time.
- GoTo (/cf1/go_to): The GoTo command consists of a global coordinate and a yaw angle. It tells the UAV where it should fly next. If the UAV has reached its location, it remains hovering at that position.
- Emergency (/cf1/emergency): The emergency command immediately turns off the motors.
- Parameter Interface: The Crazyflie server has an interface to the firmware parameters of the UAV. It provides the possibility to change them like any other ROS parameter.
- Logging Interface (/cf1/pose): The Crazyflie server allows the monitoring of the internal state variable of the UAV. With it, the UAV's assumed pose can be read and recorded.
- External Pose/Position (/poses): The Crazyflie server has an interface to send external pose or position measurements to the Crazyflie. The onboard Kalman filter integrates them in its pose estimation.

As previously mentioned, ROS is not directly running on the Crazyflie. Crazyswam is used to send high-level control commands that are handled by the onboard firmware.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

3.2 Localization

Section 2.5 introduced the difference between absolute and relative localization. In this work, two forms of absolute localization are used: motion tracking using optical sensors and reflective markers on the ground vehicle and ultra-wideband radios that measure distances between each other and use them to estimate the global position of the UAV.

Additionally, we use relative localization to determine the position of the UAV in relation to the ground vehicle. This is critical for tasks such as autonomous landing, as it allows for accurate landing and avoids missed landings, collisions, and incorrect control actions.

Before the relative localization estimation from the camera can be used, it must be converted into the global frame of reference. This conversion is achieved by applying the camera's translation and rotation transformations of the global frame of reference. This translation and rotation are obtained by combining the ground vehicle's position and rotation from the global positioning system (OptiTrack) with the constant translation and orientation that describe the camera's mounting position on the ground vehicle.

Challenges in relative localization

Coordination between the UAV and the ground vehicle is important while determining the corresponding next best control actions, and accurate relative localization plays a significant role in achieving it. However, this is a challenging problem due to the environment's dynamic nature, unreliable measurements, and unpredictable external factors. These difficulties make achieving precise relative localization between them challenging:

- **Dynamic Environment**: The UAV and the ground vehicle are not static and moving. Their pose is constantly changing.
- **Hardware Limitations**: Lightweight UAVs can only support minimal additional hardware.
- **Costs**: High-precision localization hardware is often very costly.
- **Latency**: If the data transmission is delayed or the calculation time is too long, the information can already be outdated when it is ready to use.

Potential Solutions

For our scenario, there are several possibilities to improve the UAV's localization during landing:

- **Vision-Based Systems**: Cameras can detect markers or patterns on the ground vehicle or the UAV itself, providing a visual reference that can be tracked in real-time.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- **Sensor Fusion**: Data from multiple sensors like gyroscopes, accelerometers, and magnetometers can be combined to improve localization accuracy.
- **UWB Systems**: UWB radios can be placed on the UAV and the ground vehicle to measure the distance between them.

This work explored a vision-based system as an additional relative localization system. The main advantage of using a camera is that the hardware is cheaper than other alternatives, and it can already provide a relative pose instead of just distance measurements.

Since our UAV is very lightweight and has limited computing power, we decided to mount the camera on the ground vehicle instead of the UAV (See Figure 3.1). This has the disadvantage that the camera is only beneficial for the UAV in its immediate vicinity of the ground vehicle.

The camera is not oriented vertically upwards but at a certain angle. This has the advantage that the range in the plane in which the camera can capture the UAV is larger, provided that the UAV is located in the direction in which the camera is pointed. Since we control the UAV, we can use this to our advantage by planning the landing approach based on the camera's line of sight.

A camera with a field of view (FOV) of 80° and a fish-eye lens with an FOV of 160° were tested. The significant advantage of the fish-eye camera is the broad field of view. However, with the same resolution, the camera with a standard 80° lens delivers better detection performance at a greater distance since objects in a smaller field of view appear on more pixels. Since we determine the landing approach, we do not have to rely on the advantages of a larger FOV.

To detect the position of the UAV in the image, we have resorted to detecting fiducial markers. Three approaches were explored as to how markers can be attached to the UAV:

- A single marker, which is attached to the underside of the UAV, has the same footprint as the central circuit of the UAV (35mm side length) such that it is not in the way of the propellers and does not interfere with the airflow. (See Figure 3.4)
- A cube of markers attached to the bottom of the UAV. The markers are attached to five sides of the cube (bottom and all around; the top side is used to attach to the UAV). Each face of the cube has the same footprint as the central circuit (35mm side length). (See Figure 3.5)
- A cube with markers, but without the top and bottom faces (65mm side length). This allows us to make the cube bigger with minimal influence on the airflow of the propellers. (See Figure 3.6)

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

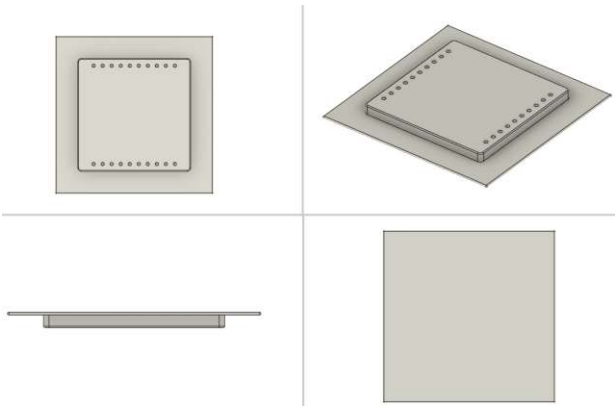


Figure 3.4: The designed attachment for the single marker, that is 3D printed.

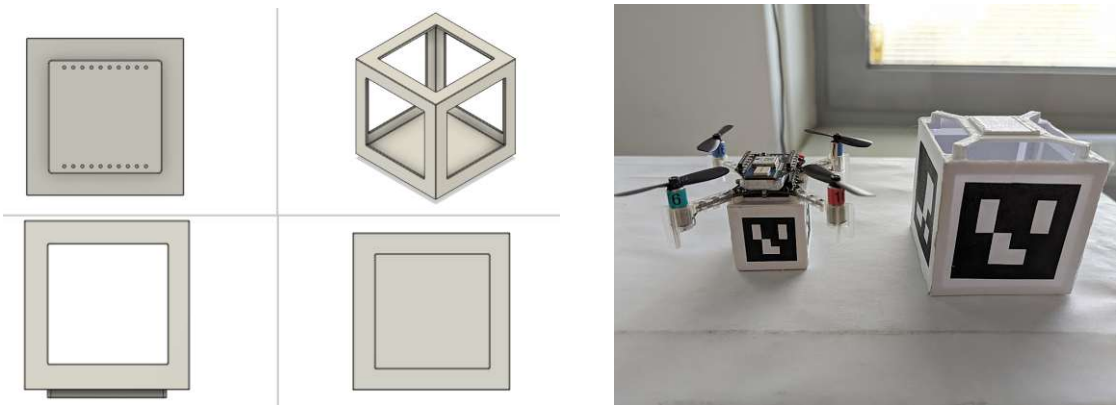


Figure 3.5: The designed attachment for five markers to form a cube that is 3D printed.

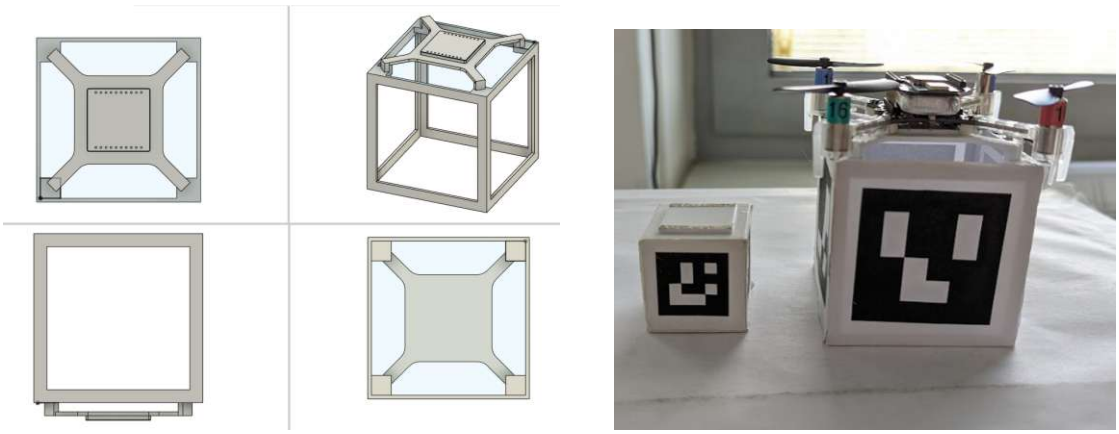


Figure 3.6: The designed attachment for four markers to form a cube (without the bottom) that is 3D printed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The cube's framework and the attachment for the single marker are designed in Fusion360 [3] and 3D printed using polylactide (PLA).

There exist multiple types of fiducial markers. We choose ArUco markers (See Chapter 2.5.1) with the smallest dictionary (4X4_50).

The landing platform is cut out of cardboard and attached to the vehicle. Our goal is not to be able to perform multiple takeoffs and landings in succession. For this, it would be necessary to realign the UAV using a robot arm or other mechanisms. This would be desirable but exceeds the scope of this work. For this work, it is sufficient if the UAV is inside the platform after landing, even if it is not straight and centered.

3.3 Feasibility

We used simulation to check the feasibility of our approach. In the following, we first provide an overview of the simulation software used and then discuss our findings.

3.3.1 Gazebo

Gazebo [30] is an open-source 3D robotic simulation environment supporting multiple robots. It is designed to simulate an environment with physics-based dynamics and can model sensors and actuators. Since the simulator can render realistic light sources, shadows, and textures, it can integrate visual sensors like cameras.

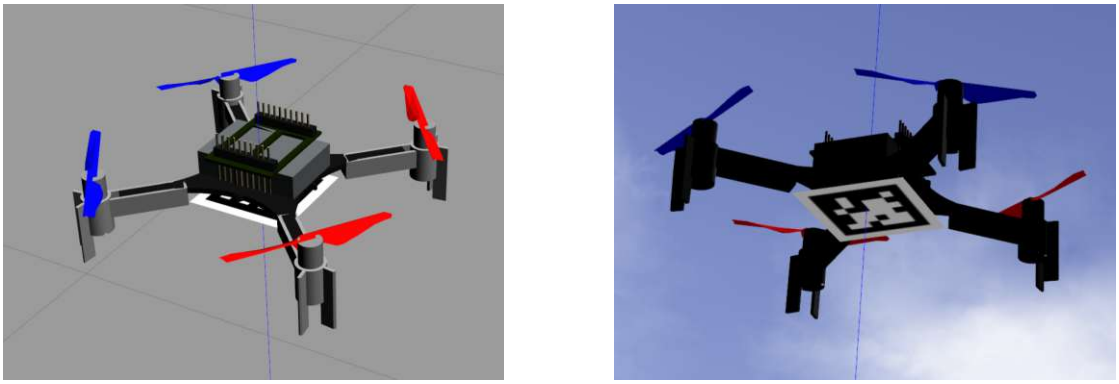
3.3.2 CrazyS

CrazyS [47] is an extension of the modular UAV simulation framework, RotorS [16], which uses Gazebo to simulate internal UAV states and behaviors like its control algorithms and state estimation, to mimic real-world conditions. The critical contribution of CrazyS is the modeling and integration of the Crazyflie 2.0 nano-quadcopter, which allows for a detailed and realistic simulation of the quadcopter's behavior in various environments. In the simulator, the UAV can be controlled by simply providing the target position to which it flies. This enables us to use the same control algorithm in software simulation and real hardware.

3.3.3 F1Tenth Simulator

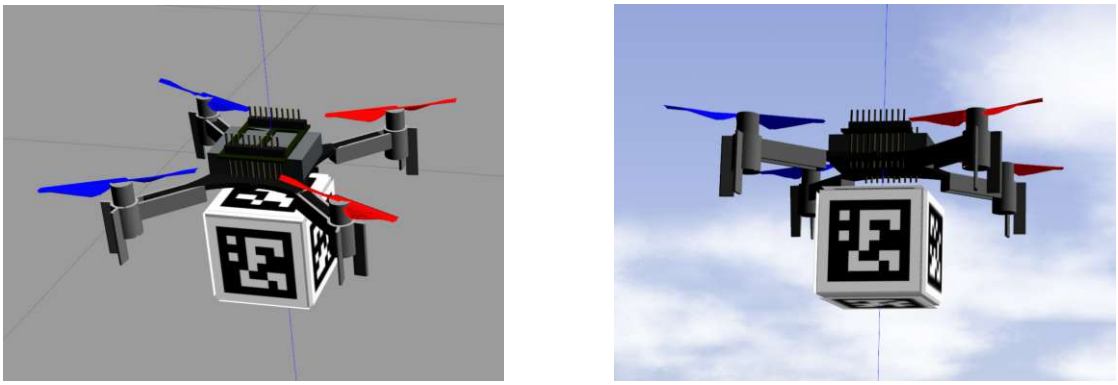
The F1Tenth Simulator by Babu et al. [4] uses Gazebo as the back end to simulate the real-world performance and behavior of an F1Tenth autonomous race car. It provides a parameterized model of the F1tenth race car, which can be controlled via the same ROS interface as the actual car. This allows a seamless transition of algorithms from the simulation to the real world.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



(a) top view (b) bottom view

Figure 3.7: The UAV with an attached marker in the simulation environment Gazebo.



(a) top view (b) bottom view

Figure 3.8: The UAV with an attached marker cube in the simulation environment Gazebo.

3.3.4 Simulation Setup

The simulation uses the ROS packages for CrazyS and the F1Tenth Simulator and the native ROS package RVIZ [29] for visualization. The provided model of the F1tenth race car was extended to include the landing platform and the camera to detect the UAV. The Crazyflie model was extended to allow attaching either a single marker or a cube of markers for UAV detection. Both the extended model of the car and the extended model of the Crazyflie UAV can be seen in Figure 3.7, Figure 3.8, and Figure 3.9.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

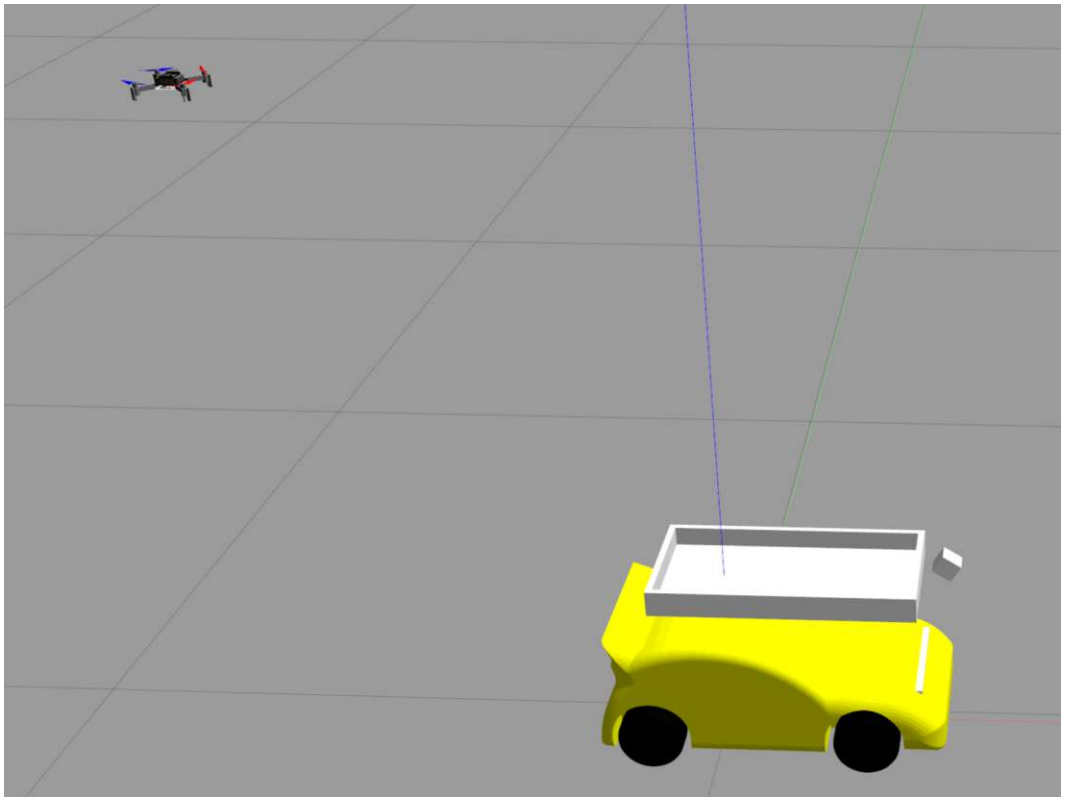


Figure 3.9: The UAV and the ground vehicle in the simulation environment Gazebo

3.3.5 ArUco detector

The detection of the ArUco markers is performed using the OpenCV library [11]. The package *ros_aruco_opencv* [49] developed by B. Sowa acts as a ROS wrapper to the OpenCV library that listens to the ros camera topics and calls the necessary OpenCV library functions for the pose estimation.

3.3.6 Findings

The simulation shows that all three approaches are viable, as the mobile vehicle can recognize the marker's position. However, the cube detection is much more reliable than just one single marker, as at least one marker is visible from various angles and is, therefore, also preferable.

It also shows that the cube's size significantly influences the detection range, which makes a big cube always preferable. However, subsequent testing with hardware shows that a big cube also increases the weight and air resistance during flying, which, in turn, limits the optimal size of the cube.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

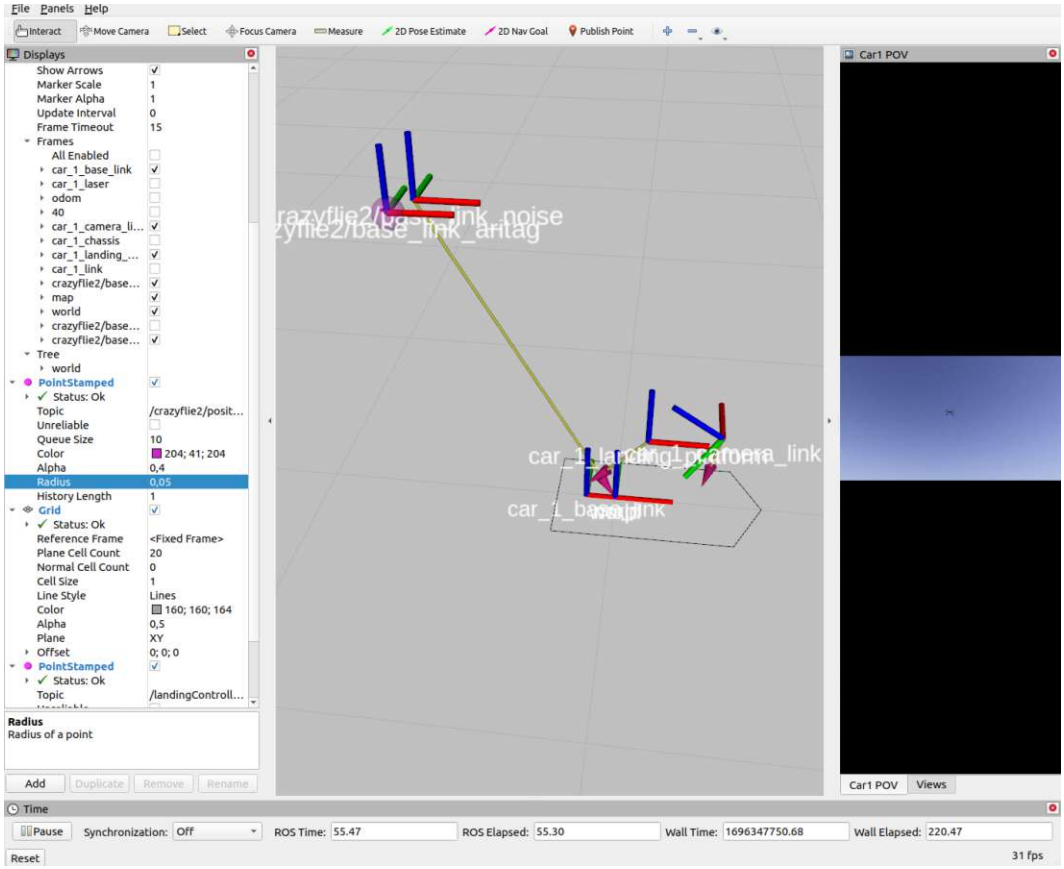


Figure 3.10: Screenshot of RVIZ, showing the camera image, the vehicle positions, and the position estimation of the camera

This led us to a final iteration of the marker cube (See Figure 3.11) with a side length of 45 mm. In addition, lightweight construction was essential to keep the weight as low as possible.

3.4 Integration of relative Position Measurements

Now that we have established how to achieve relative position measurements, we must incorporate them into the Crazyflie. Since its firmware already does sensor fusing of the UWB, accelerometer, and gyroscope measurements using an EKF, it also makes sense to use it for the relative position data. It is essential to use an adequate noise model so the system knows to what extent the new position data is relevant, as also described in [35], we model the noise in simulation.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

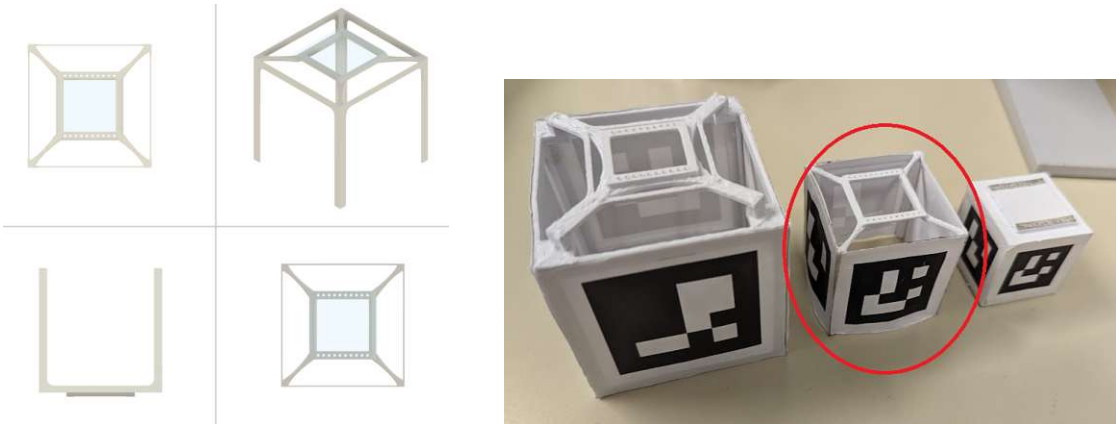


Figure 3.11: The designed attachment for four markers to form a cube (without the bottom, extra lightweight) that is 3D printed.

3.4.1 Measurement recording

The noise model describes the variance of each measurement, which the EKF needs for the update step.

To record the variance of the ArUco marker detection, we again rely on our simulation environment, as it provides the ground truth position of each object in the simulation.

We let the UAV (with attached marker cube) and the ground vehicle drive around randomly. Particular attention was paid to ensuring that the UAV approaches the camera from as many different directions and angles as possible.

Simultaneously the following is recorded for each measurement $\mathbf{z}_t$:

- The position of each marker p_{aruco}, calculated from the ArUco node.
- The ground truth position p_{gt}, reported by the simulation environment, transformed in the camera frame.
- The actual distance between the marker and the camera d.
- The angle between the marker's normal vector and the camera's normal vector ρ.

Both distance d and angle ρ are parameters for the recorded noise. They are chosen since both the distance and the angle play a significant role in the relevance of the measurement. The further away the marker was at the time of detection, the more difficult it was to obtain an accurate position. A similar assumption about the angle can also be made. Their respective influence is quantified by the following.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

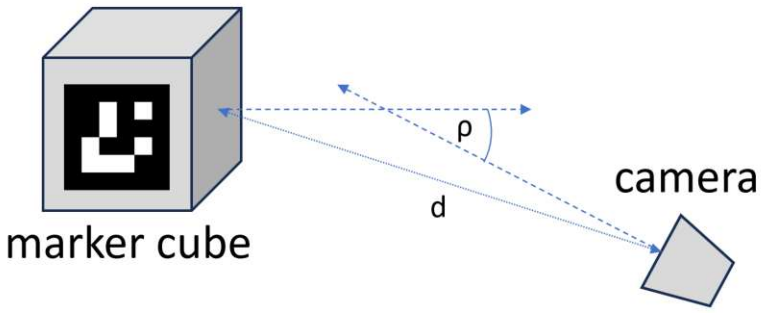


Figure 3.12: A graphical representation of the noise model parameters d and ρ and how they relate to the marker-cube and the camera: d is the distance between the marker and the camera, ρ is the angle between the normal vectors of the marker and the camera.

The error is defined in the following way:

$$e = \begin{bmatrix} e_x \\ e_y \\ e_z \end{bmatrix} = |p_{gt} - p_{aruco}| = \begin{bmatrix} |x_{p_{aruco}} - x_{p_{gt}}| \\ |y_{p_{aruco}} - y_{p_{gt}}| \\ |z_{p_{aruco}} - z_{p_{gt}}| \end{bmatrix} \tag{3.1}$$

Figure 3.13 shows the recorded error in relation to both d and ρ for each axis.

All measurements are grouped in a fixed grid where each cell's variance σ is calculated using the following formula if there are at least ten measurements inside of it:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{3.2}$$

with

$$\mu = \frac{1}{n}\sum_{i=1}^{n}(x_i), \tag{3.3}$$

n being the number of items in each cell and x_i being the according value out of e_x, e_y and e_z respectively. The result can be seen in Figure 3.14.

The data indicate that variance positively correlates with the distance d and, excluding the y-axis, it also depends on angel ρ. Quantitatively, the effect of distance on variance is significantly more pronounced than the influence of angle. Furthermore, the variance associated with the z-axis exceeds that of the other axes. This observation can be attributed to the z-axis representing depth information, which is challenging to estimate with an ArUco estimator.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

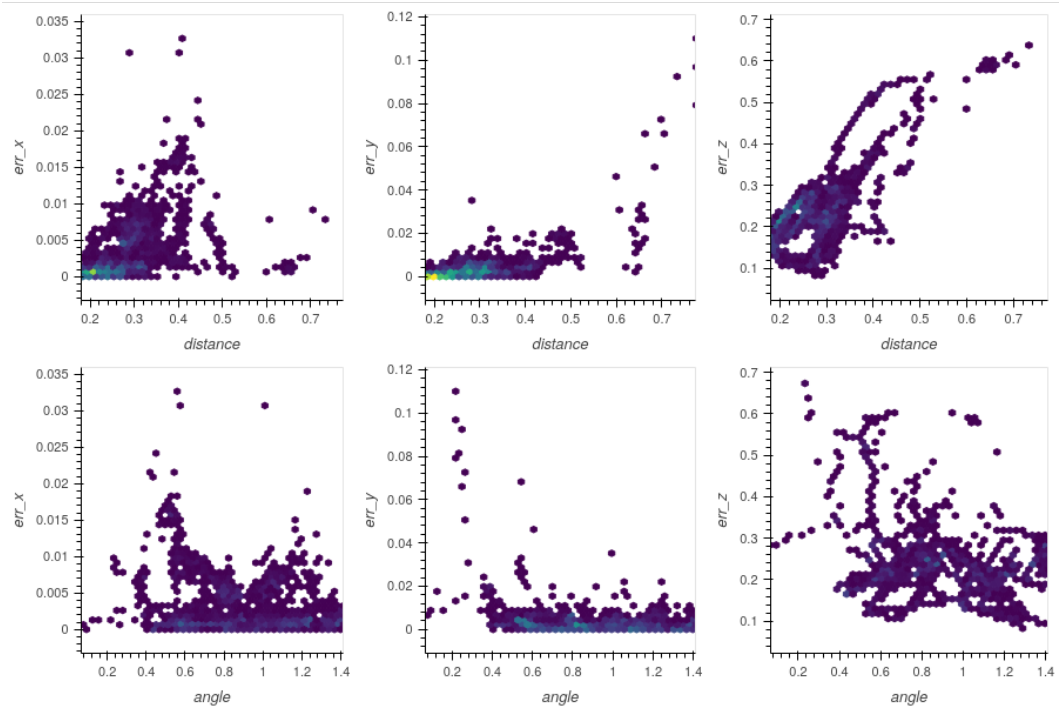


Figure 3.13: The measurement error. The top three diagrams show the error in relation to the distance d; the bottom three diagrams show the error in relation to the angle ρ.

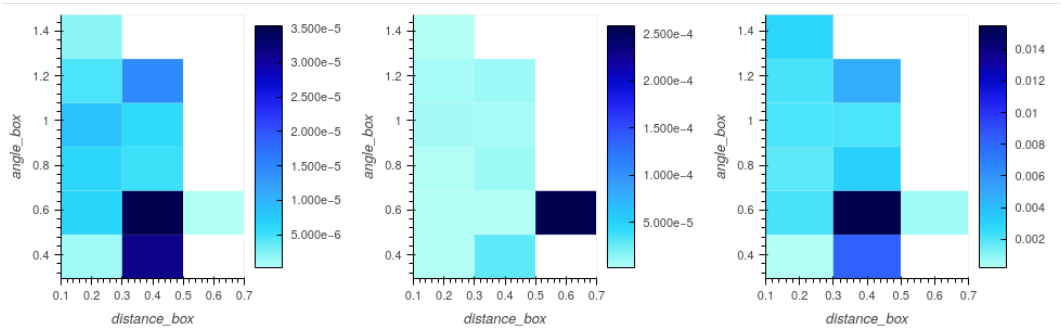


Figure 3.14: Heatmap of the calculated variance grid. From left to right, the heatmap for σ_x^2, σ_y^2 and σ_z^2.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

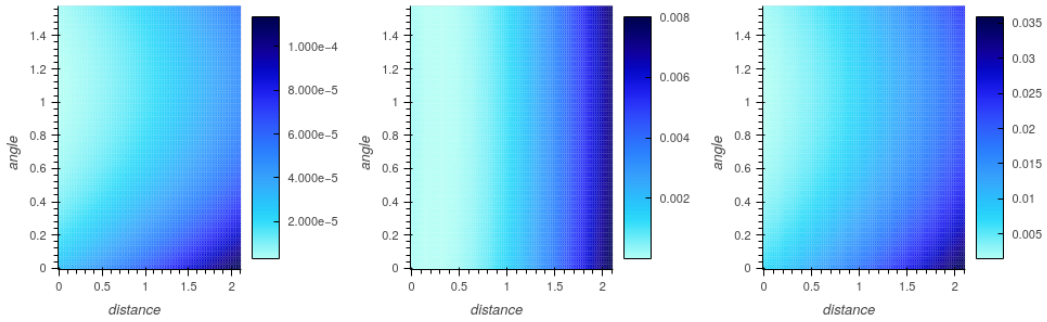


Figure 3.15: The fitted noise model. From left to right, $\sigma_x^2(d,\rho)$, $\sigma_y^2(d,\rho)$ and $\sigma_z^2(d,\rho)$.

An important observation is that the variance depending on angle is not, as assumed, lowest at 0 rad, but at approximately 0.9 rad. This corresponds to the orientation at which the UAV is upright. If the UAV moves too quickly and is therefore no longer oriented vertically, the position detection accuracy decreases.

3.4.2 Noise Model

We obtain the noise model by finding the correct approximation of the observed variance. We will do this independently for each axis.

Deciding on a function with which the variance can be approximated leaves a lot of leeway. Based on the measured values, we have chosen the following function:

$$\sigma^2(d,\rho) = (p_a + p_b p_c^{p_d d + p_e} + p_f \rho + p_g \rho^2)^2 \tag{3.4}$$

It combines an exponential function of d and a quadratic function of ρ. It is squared to make sure the result is positive.

The necessary parameters p_a, p_b, p_c, p_d, p_e and p_f are obtained by fitting the function with *optimize.curve_fit()* from the python library *SciPy* [54]. This results in the approximation seen in Figure 3.15.

To account for real-world inaccuracies not present in the simulation environment, like the imprecise transformation of the position measurements from the camera frame to the world frame, a constant factor k_{system} is added to p_a.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

3.4.3 Implementation details

ArUco detector

As in the simulation, ArUc markers are detected using the *aruco_opencv* ROS package [49]. The base implementation was extended to perform the following additional tasks:

- **Cube detection**: The node can detect single ArUco markers and ArUco boards that consist of multiple ArUco markers. Each detected marker contributes to the board's single pose estimation, making it more accurate and allowing pose estimation in the presence of occlusions. Only 2D boards are supported by *aruco_opencv*, so the capability of defining a 3D board in the shape of a cube has been added.
- **Interface to crazyswarm2**: The resulting pose of the detected cube has to be sent to the UAV; hence, the ability to interface with the crazyswarm2 node using the topic /positions was added.

With our extension, the positions of multiple detected markers are combined using the detect board function of OpenCV and transformed into the world frame.

Let $\mathbf{p_{camera}}^{W}$ be the position vector of the camera in the world frame and $\mathbf{R_{camera}}^{W}$ be the rotation matrix of the camera in the world frame. Then, the position of the cube in the world frame is

$$\mathbf{p^W_{cube}} = \mathbf{p^W_{camera}} + \mathbf{R^W_{camera}} \mathbf{p^C_{cube}} \tag{3.5}$$

With $\mathbf{p^C_{cube}}$ as the cube's position in the camera frame.

The variance is estimated for each marker using the noise model and combined into a single variance. For this, we exploit the following property of the sum of Gaussians:

$$\sigma^2 = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}} = \frac{1}{\sigma_1^{-2} + \sigma_2^{-2}} \tag{3.6}$$

Like with the position, the variance must also be transformed into the world frame. This is done using the following:

$$\Sigma^W_{cube} = \mathbf{R^W_{camera}} \Sigma^C_{cube} \mathbf{R^{W\top}_{camera}} \tag{3.7}$$

with $\mathbf{R^{W\top}_{camera}}$ being the transpose of $\mathbf{R^W_{camera}}$ and

$$\Sigma^C_{cube} = \begin{bmatrix} \sigma_x^2(r,\rho)^C & 0 & 0 \\ 0 & \sigma_y^2(r,\rho)^C & 0 \\ 0 & 0 & \sigma_z^2(r,\rho)^C \end{bmatrix} \tag{3.8}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Modifications to crazyswarm

The Crazyswarm2 framework provides an extensive interface to communicate with the Crazyflie UAV. However, it does not allow continuous transmission of external position data with variable variance as it only has an interface for global positioning systems with constant variance, such as motion capture.

We extended this missing function by implementing an additional interface to send position data, including the corresponding variance. This also required several extensions to the underlying *crazyflie_cpp* library [24], which is responsible for the low-level communication with the Crazyflie.

This new position data is sent using the topic /positions.

Modification to the onboard firmware of the Crazyflie

The Crazyflie firmware was extended to handle the custom external position messages with corresponding confidence values. Suppose the position data comes from our new message interface in the update step. In that case, the Kalman filter performs a scalar update for each axis using the given variance instead of the predefined static one.

3.5 Landing controller

In the remainder of this chapter, we turn to our second objective, designing a high-level landing controller.

The landing_controller node serves as the high-level controller of the Crazyflie UAV and is developed in ROS. It then uses position data from the ground vehicle and the UAV to control the UAV by sending target coordinates where it should fly. It has the following inputs and outputs.

Inputs:

- /cf1/pose: The pose of the UAV.
- /autonomous_landing/car_state: The state variables of the car, estimated by the UKF.
- /joy: The button presses on the joystick.
- /ackermann_cmd: The current velocity and angle command controls the ground vehicle.
- /autonomous_landing/takeoff: Alternative input to trigger the takeoff without the joystick.
- /autonomous_landing/land: Alternative input to trigger the landing without the joystick.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- /autonomous_landing/follow: Alternative input to trigger the following command without the joystick.

Outputs:

- /cf1/takeoff: Interface to Crazyswarm to takeoff.
- /cf1/land: Interface to Crazyswarm to land.
- /cf1/go_to: Interface to Crazyswarm to fly to a specified position.
- /cf1/emergency: Interface to Crazyswarm to immediately stop the UAV's motors.
- /autonomous_landing/car_prediction: The future ground vehicle position prediction for visualization and evaluation.
- /autonomous_landing/landing_path: The current landing trajectory that the UAV tries to follow during the landing procedure.

The control behavior depends on the current state of the controller, which can be one of the following states:

- **Inactive**: No new target coordinates are sent to the UAV. If the UAV has already performed its last command, it simply holds its position.
- **Following**: The UAV follows the ground vehicle at a constant height in the following state.
- **Landing**: In the landing state, the controller initiates the landing procedure, navigating the UAV to the ground vehicle to land on its landing platform.

The controller begins to function with the takeoff, which initiates the inactive state. Joystick button presses trigger a transition. Figure 3.16 shows the internal state machine of the controller.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

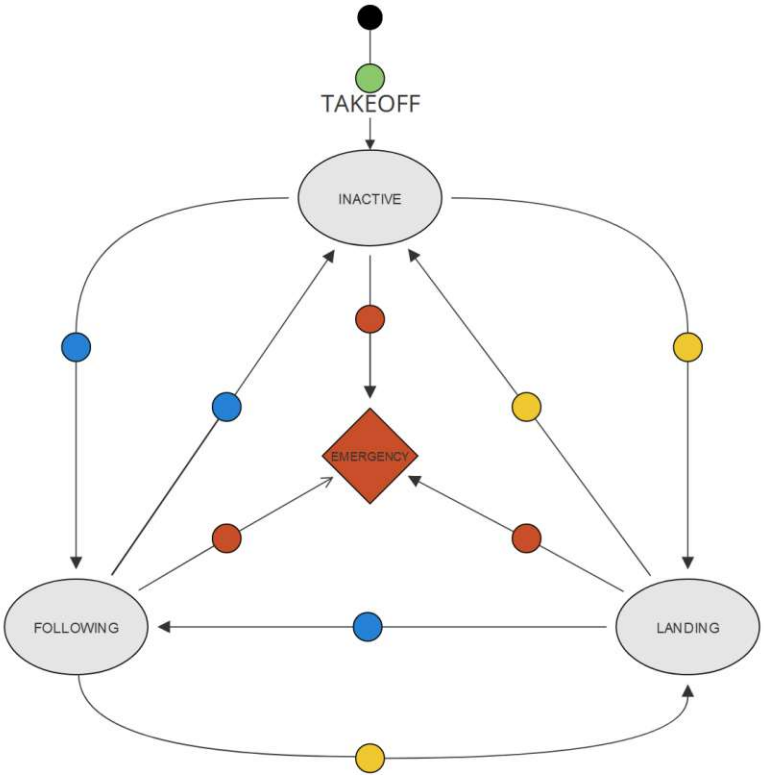


Figure 3.16: The controller's state machine, showing the state changes with each button press shown as colored dots in the figure.

3.5.1 Prediction of the landing target

To predict the future position of the ground vehicle, the kinematic single-track model is used (as described in [1]). It defines the following state variables (See Figure 3.17):

- s_x: The x coordinate of the ground vehicle's position, with the rear wheel as the reference point.
- s_x: The x coordinate of the ground vehicle's position, with the rear wheel as the reference point.
- δ: The cars steering angle.
- v: The velocity of the ground vehicle in its body frame.
- Ψ: The heading angle (i.e. the orientation in the x/y-plane) of the ground vehicle.
- a: The acceleration of the ground vehicle in its body frame.
- v_δ: The steering velocity.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

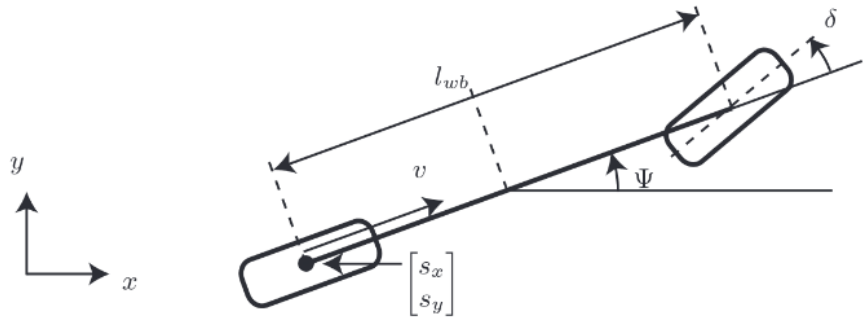


Figure 3.17: Kinetic single-track model, taken from [1].

And the derivatives:

$$\dot{s}_x = v \cos \Psi \tag{3.9}$$

$$\dot{s}_x = v \sin \Psi \tag{3.10}$$

$$\dot{\delta} = v_\delta \tag{3.11}$$

$$\dot{v} = a \tag{3.12}$$

$$\dot{\Psi} = \frac{v}{l_{wb}} \tan \delta \tag{3.13}$$

The ground vehicle state variables are estimated using a UKF (See Section 2.6.3), implementing the same kinematic single-track model. The state estimator was implemented as a separate ROS node using the *filterPy* [25] library. The position and orientation measurements were taken from OptiTrack, the acceleration was measured by the built-in accelerometer of the speed controller, and the steering angle control input was used as the steering angle measurement.

The single-track model predicts the ground vehicle's position at time $t_{meeting}$. We derive $t_{meeting}$ from a simplified 1D approximation of the distance d between the ground vehicle and the UAV:

$$d = -v_{diff} t + d_0 \tag{3.14}$$

With $v_{diff} = v_d - v_c$, v_d being the UAV's maximal flying velocity, v_c being the current speed of the ground vehicle (assuming $v_d \gg v_c$), and d_0 being the Euclidean distance between the vehicle and the UAV.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

If we now set $d = 0$ and solve Equation 3.14 for t we get the approximated meeting time $t_{meeting}$:

$$t_{meeting} = \frac{d_0}{v_{diff}} \tag{3.15}$$

In the last step, the static transform between the ground vehicle and the landing platform is applied to the predicted ground vehicle position to get the expected landing platform position.

3.5.2 Landing trajectory generation

The landing trajectory is the path the UAV tries to follow during the landing process. We have defined three properties that our landing trajectories need to have to guarantee safe landings and improved camera detection:

- **Smoothness**: A smooth landing trajectory avoids abrupt changes in directions or speed, which prevents unstable flight dynamics.
- **Verticality**: The landing trajectory should ensure that the UAV descends from above the landing platform.
- **Orientation**: The UAV should assist the detection framework by approaching the ground vehicle in the FOV of the camera.

Bezier curves were first introduced by P. Bézier and are widely used in the fields of design and computer graphics. The curve is smooth by design, and using control points, the shape of the curve can be adjusted to satisfy specific path constraints. Based on a set of control points, the Bezier curve can be calculated deterministically, making it ideal to use them as landing trajectories since they also satisfy all of our defined properties.

The first control point corresponds to the beginning of the Bezier curve, while the last control point corresponds to the end of the curve. Therefore we defined the UAV's current position to act as the initial control point and the future expected position of the landing platform functions as the final control point.

The control points in between differ depending on the positioning of the UAV in relation to the ground vehicle and do not align with the resulting curve:

- If the UAV is very close to the ground vehicle and within a specific distance threshold, it will switch to a descent mode, moving directly toward the landing platform. Suppose this descent mode is started and engaged. In that case, it will not leave this mode even if the distance threshold is exceeded, preventing erratic behavior or oscillations in the UAV's trajectory and ensuring a stable and consistent landing approach.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

- If the UAV is in view of the camera, the ArUco tags can be detected. As a result, increased accuracy of the UAV's position data can be assumed to satisfy our orientation property. To satisfy verticality, an additional control point directly above the predicted position of the landing platform is introduced. The resulting Bezier curve ensures the UAV approaches the landing platform at the correct height (See Figure 3.18).
- If the UAV is not in view of the camera, its positioning data has a high inaccuracy. Therefore, flying into the FOV of the camera is the priority. Since the camera faces backward, an additional control point is set behind the ground vehicle. The resulting Bezier curve ensures that the UAV, regardless of where it is currently located, always approaches from behind the ground vehicle first within the camera's FOV (See Figure 3.19).

3.5.3 Field Of View Check

The camera mounted on the ground vehicle detects the UAV and increases the accuracy of the positioning data. For the UAV to be seen, it must be within the camera's FOV. Therefore, it is essential to determine whether the UAV can theoretically be detected, regardless of whether it is seen. As explained in [22], this can be efficiently calculated:

For this, we transform the UAV's position in the camera's coordinate frame where the camera is located at the position $(0, 0, 0)$ with the lens pointed in the positive z-direction. Let's define the position as the point $\vec{p}$ and the unit vector along the line of sight as $\hat{n}$.

For the point $\vec{p}$ to be visible, it must be in front of the camera. Therefore, the following must hold:

$$\vec{p} \cdot \hat{n} = p.z > 0 \tag{3.16}$$

Now, we project the 3D coordinate $\vec{p}$ on a plane at an arbitrary distance $d > 0$.

The projected coordinate $\vec{p'}$ is defined by the intersection of the projection plane and the ray along the vector $\vec{p}$. It is calculated the following way:

$$\vec{p'} = \frac{d\vec{p}}{\hat{n} \cdot \vec{p}} \tag{3.17}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

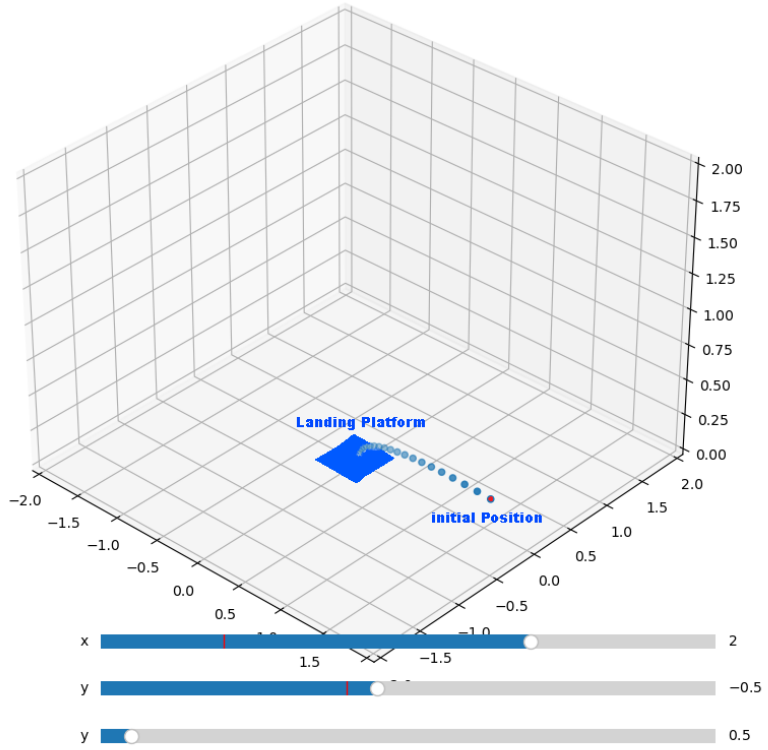


Figure 3.18: The Bezier curve generated with the UAV inside of the FOV of the camera (Bezier with three control points). As it can be seen, the UAV flies towards the landing platform, coming from above.

The projection plane is bounded in both x- and y-direction by the bound b_x and b_y, depending on the FOV of the camera. Assuming the FOV α is symmetrical (i.e $b = b_x = b_y$), the following relation holds:

$$\tan(\alpha/2) = \frac{b}{d} \tag{3.18}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

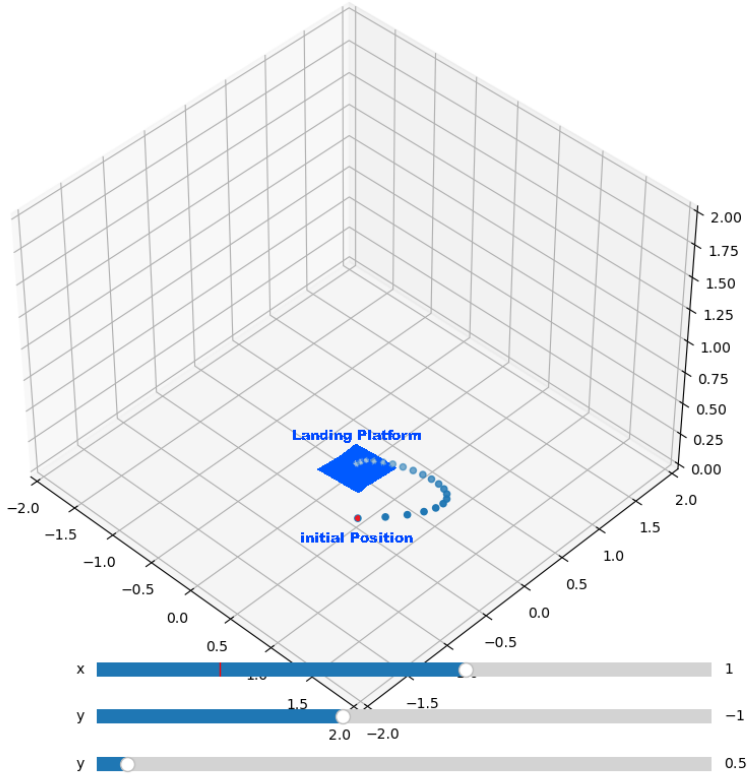


Figure 3.19: The Bezier curve generated with the UAV outside of the FOV of the camera (Bezier with four control points). As can be seen, the UAV does not fly directly to the landing platform; it will first try to get in the camera's view.

If $p'.x$ and $p'.y$ lie inside the boundary b, it is inside the FOV of the camera. Using Equation 3.18, the following conditions must be satisfied:

$$|p'.x| \leq d \tan(\alpha/2) \tag{3.19}$$
$$|p'.y| \leq d \tan(\alpha/2) \tag{3.20}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

By setting $d = 1$, the conditions from Equation 3.19 and 3.20 can be simplified to the following:

$$\left|\frac{p.x}{p.z}\right| \leq \tan(\alpha/2) \tag{3.21}$$

$$\left|\frac{p.y}{p.z}\right| \leq \tan(\alpha/2) \tag{3.22}$$

Together with Equation 3.16, they can be used to check efficiently if a coordinate lies inside the FOV of the camera.

3.5.4 Trajectory following

While in *landing mode*, the controller constantly generates a landing trajectory based on the current position of both the UAV and the ground vehicle. The UAV tries to follow the trajectory using a version of the path tracking algorithm *pure pursuit*.

A point on the trajectory is selected based on its Euclidean distance to the UAV, which is nearest to the specified *lookahead distance* parameter, denoted as L. This point is then sent to the low-level set-point controller of the UAV.

As *yaw*, the orientation of the UAV in the x/y-plane is set to the same value as the yaw of the ground vehicle.

3.6 Software Overview

Figure 3.20 acts as a summary of the system design by showing the overview of the software setup.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

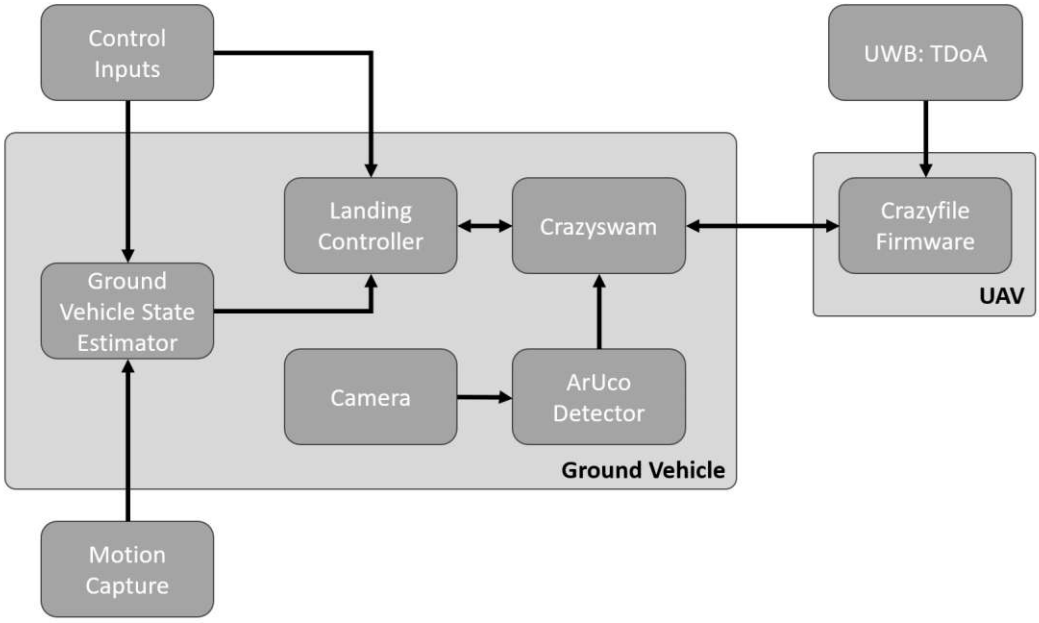


Figure 3.20: High-level overview of how the different parts of the software setup are connected to each other

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Evaluation

Various tests using the actual hardware evaluated the system's performance. The motion capture system provided the ground truth for the UAV and the ground vehicle. The available makers for the UAV are attached on top using the *mocap marker deck* from Bitcraze (See Figure 4.1).

Because of the UAV's small footprint, the motion tracking marker also needed to be smaller (compared to the markers on the ground vehicle). This resulted in poorer tracking performance, leading to irregular loss of recognition. The positioning accuracy is not affected by the use of smaller markers.

Figure 4.1: The Crazyflie UAV with attached mocap maker deck with markers.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.1 UAV Positioning

The performance of the UAV positioning system was evaluated at a standstill and while flying:

For the standstill experiment, the UAV was placed elevated at the coordinate (0.00 m, 0.00 m, 0.70 m), and the ground vehicle was placed in the y direction with the camera at the coordinate (0.00 m, 0.90 m, 0.20m) such that the UAV was in its view. While flying, the ground vehicle was not moved, and the UAV was instructed to hover at (0.00 m, 0.00 m, 1.00 m). For 4 minutes, the ground truth, as well as the position estimates, are recorded at a Frequency of 10 Hz for the following four scenarios:

- UAV position (from the UKF of the UAV), while the UAV is at a standstill, without the ArUco marker detection running *(stationary, no cam)*.
- UAV position (from the UKF of the UAV), while the UAV is at a standstill, with the ArUco marker detection running *(stationary, cam)*.
- UAV position (from the UKF of the UAV), while the UAV is flying, without the ArUco marker detection running *(flying, no cam)*.
- UAV position (from the UKF of the UAV), while the UAV is flying, with the ArUco marker detection running *(flying, cam)*.

Figure 4.2 shows the Root Mean Squared Error (RMSE) for each axis, calculated by

$$\texttt{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} ||x(i) - \hat{x}(i)||^2}{N}}, \tag{4.1}$$

with $x(i)$ as the ground truth position coming from Optitrack and $\hat{x}(i)$ as the estimated position coming from the Kalman filter of the UAV.

The results presented here demonstrate a significant improvement in positional accuracy for all axes, with the most significant enhancement noted in the X-axis.

This improvement aligns with our previous observations in Section 3.4.1, where we established that position estimation using ArUco markers is less effective for depth (Z-axis) than for lateral positions (X- and Y-axes). Given that the ground vehicle, and consequently the camera, is oriented along the Y-axis with an upward gaze, the poor depth information — as interpreted from the camera's frame of reference — has a more pronounced influence on the accuracy of both the Y- and Z-axes.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

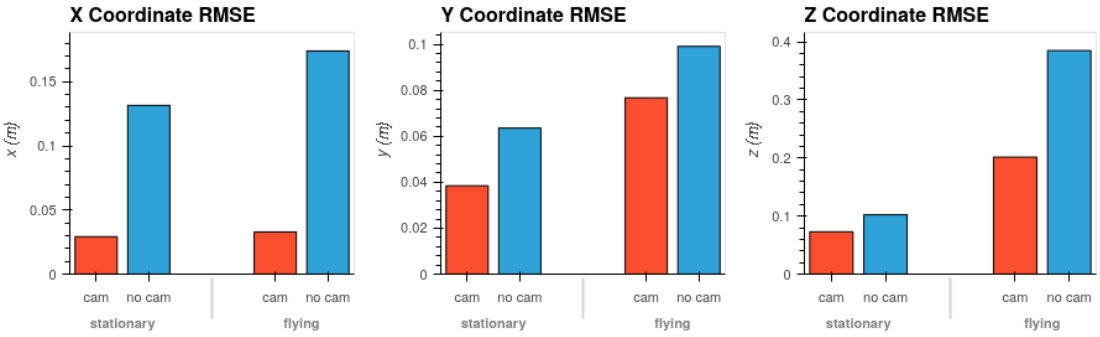


Figure 4.2: RMSE of each axis, with and without ArUco Cube detection, stationary and while flying (Hovering)

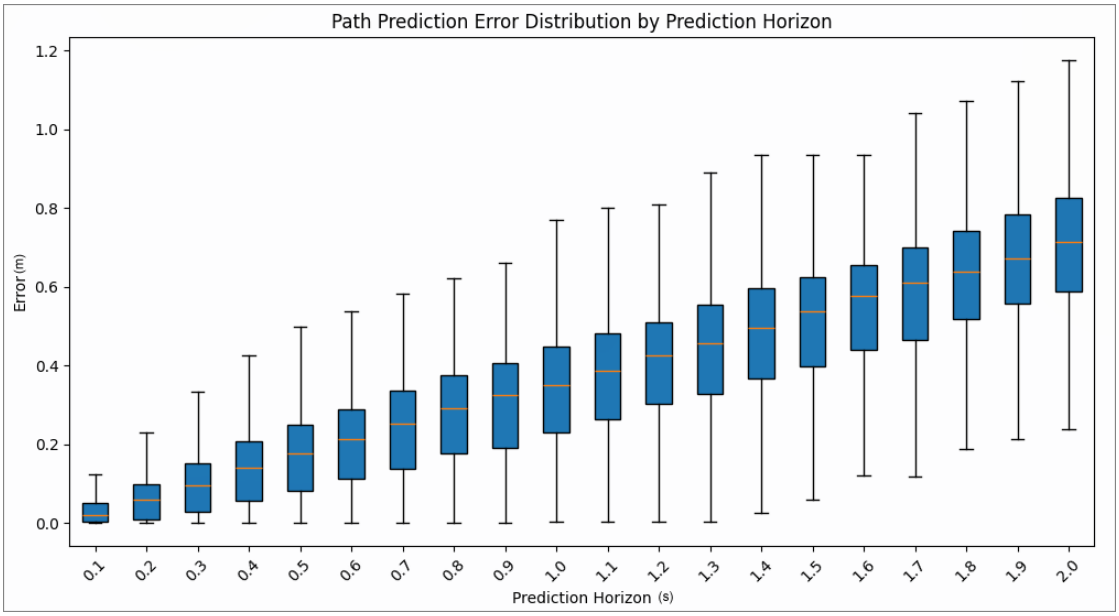


Figure 4.3: Boxplot showing the error of the landing target prediction for different prediction horizons.

4.2 Landing Target Prediction

The performance of the landing target prediction was evaluated using OptiTrack as ground truth. While randomly driving at a constant speed of $0.6m/s$, the ground vehicle's current position and predicted future position were recorded. After matching the times of the predicted position with the actual position, the error is calculated. For this evaluation, a prediction horizon of up to 2 seconds was chosen.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

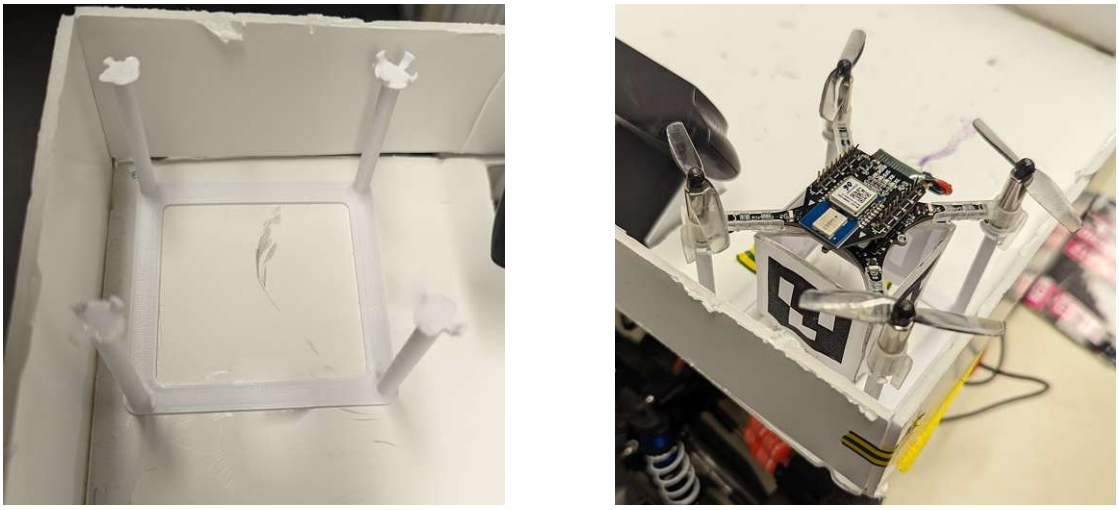


Figure 4.4: The designed UAV mounting bracket for the landing platform where the UAV rests before takeoff.

Figure 4.3 shows the prediction error for different prediction horizons as a boxplot where the box expands from the first to the third quartile with the line representing the median. It can be seen that the error increases linearly with the length of the prediction horizon.

The evaluation of the landing prediction target informs on how valid the predictions are and to which point they can be used. It can be concluded that they are usable up to a prediction horizon of 1 second. Since the controller constantly updates the target position and the prediction horizon calculated by Equation (3.15) gets smaller the closer the UAV is to the landing target, the relevance of accurate target predictions becomes less critical with higher prediction horizons.

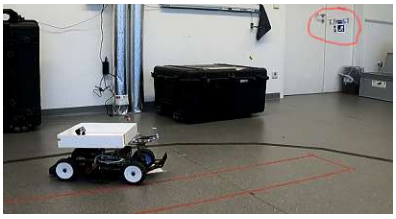
4.3 Landing Performance

The whole system's performance was tested the following way: While driving, the UAV rests on a mounting bracket on the landing platform (see Figure 4.4), where it waits for the takeoff command. Afterward, the UAV hovers in place, waiting for the landing command that initiates the landing procedure. This scenario was repeated 40 times, 20 times with the ground vehicle stationary during the landing procedure and 20 times with the ground vehicle driving at a constant velocity of 0.6 m/s. See Figure 4.5 for reference.

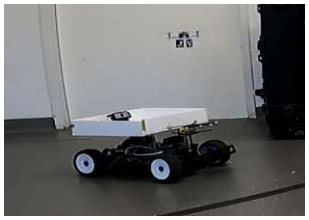
Seventeen out of the twenty attempts where the ground vehicle is stationary were successful, showing that our approach is viable. Unfortunately, a similarly high success rate could not be reported for the landing attempts where the car was driving at a constant speed. Only one out of the twenty attempts were successful.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

(a) UAV is starting from the platform of the ground vehicle.



(b) While the ground vehicle moves through the room, the UAV hovers.



(c) Landing is initiated and the UAV lands on the platform of the moving ground vehicle.

Figure 4.5: Pictures showing the different stages of the system evaluation.

We attribute these limitations mainly to the used hardware: We are limited by the used computing platform and can only run our main controller at 10 Hz and the ground vehicle state estimation at 15 Hz, making it unreliable and unable to adjust for sudden changes. Additionally, to utilize the maximal frequency of 30 FPS of our ArUco marker detection camera, we had to reduce its image size to 480p. With a faster computing platform, we expect both the detection quality and the prediction and trajectory generation to improve, making more reliable landing possible compared to our current proof of concept.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Conclusion and Outlook

In this thesis, we focused on two aspects:

- Improve the localization of the UAV by integrating relative localization data obtained from a camera on the ground vehicle.
- Design a high-level landing controller that lands on a small landing platform on the ground vehicle.

To the best of our knowledge, all current vision-based implementations of such a system use cameras attached to the UAV itself. In this work, in contrast, we proposed a system, where a camera is placed on the ground vehicle. Throughout various design iterations and evaluations in simulation and with real hardware, we chose to use ArUco markers attached to a cube at the bottom of the UAV. The position of each marker was estimated using ArUco pose estimation, which then was combined and integrated with a noise model derived from our simulation environment. To achieve this, it was necessary to adapt the detection node, communication pipeline, onboard software, and the Kalman filter of the UAV. It was shown that integrating relative positioning data significantly improves the UAV positioning compared to relying only on UWB positioning. A downside of our approach is that the improvement quality is related to the relative position between the ground vehicle and the UAV.

We furthermore designed a high-level landing controller to generate landing trajectories with required smoothness, verticality, and orientation properties, by using Bezier curves. Smoothness is important to avoid abrupt changes in direction and speed, making flights more stable and making the UAV easier for the camera to detect. With verticality, we define the trajectory as ensuring that the UAV descends from above the landing platform. Orientation combines our two goals in ensuring that the UAV approaches in the view of the camera.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The trajectories are continually generated based on the UAV's current position and the landing platform's predicted future position.

Like the trajectories, the future position of the landing platform was estimated continuously by estimating the current state of the ground vehicle using a UKF and interpolating them into the future. It was shown that the future state prediction was usable up to a prediction horizon of one second but not as reliable as needed for landing while the ground vehicle was moving.

We presented and quantitatively evaluated an implementation of both aspects. This was done using mainly open-source hardware and software, which can be adapted easily for similar or other applications in field robotics.

When evaluating this landing controller, we noticed several limitations that mainly arise from the used hardware.

5.1 Future Work

As the current solution is not as reliable as needed to land while the ground vehicle is moving, there are still some possibilities for improvement. With a faster computing platform, we expect both the detection quality and the prediction and trajectory generation to improve, making reliable landing possible, as our proof of concept showed. Therefore it might be worth for future work to test our approach on different hardware platforms and configurations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

List of Tables

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Algorithms

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] Althoff, Matthias ; Koschi, Markus ; Manzinger, Stefanie: CommonRoad: Composable benchmarks for motion planning on roads. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, S. 719–726

[2] Araar, Oualid ; Aouf, Nabil ; Vitanov, Ivan: Vision Based Autonomous Landing of Multirotor UAV on Moving Platform. In: *Journal of Intelligent & Robotic Systems* 85 (2017), Feb, Nr. 2, 369-384. http://dx.doi.org/10.1007/s10846-016-0399-z. – DOI 10.1007/s10846–016–0399–z. – ISSN 1573–0409

[3] Autodesk: *Fusion 360*. https://www.autodesk.de/products/fusion-360/. – visited on 2024-01-07

[4] Babu, Varundev S. ; Behl, Madhur: f1tenth. dev-An Open-source ROS based F1/10 Autonomous Racing Simulator. In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)* IEEE, 2020, S. 1614–1620

[5] Barry, Richard: *FreeRTOS*. https://www.freertos.org/index.html. – visited on 2024-01-07

[6] Bay, Herbert ; Tuytelaars, Tinne ; Van Gool, Luc: SURF: Speeded Up Robust Features. (2006), S. 404–417. ISBN 978–3–540–33833–8

[7] bitcraze: *cflib: Crazyflie python library*. https://github.com/bitcraze/crazyflie-lib-python. – visited on 2024-01-07

[8] bitcraze: *Getting started with the Crazyflie 2.X*. https://www.bitcraze.io/documentation/tutorials/getting-started-with-crazyflie-2-x/. – visited on 2024-02-11

[9] bitcraze: *Stabilizer Module*. https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/. – visited on 2024-01-07

[10] bitcraze: *Two Way Ranging Protocol*. https://www.bitcraze.io/documentation/repository/lps-node-firmware/master/protocols/twr-protocol/. – visited on 2024-01-06

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[11] Bradski, G.: The OpenCV Library. In: *Dr. Dobb's Journal of Software Tools* (2000)

[12] Chao, HaiYang ; Cao, YongCan ; Chen, YangQuan: Autopilots for small unmanned aerial vehicles: A survey. In: *International Journal of Control, Automation and Systems* 8 (2010), Feb, Nr. 1, 36-44. http://dx.doi.org/10.1007/s12555-010-0105-z. – DOI 10.1007/s12555–010–0105–z. – ISSN 2005–4092

[13] Chen, Yunfei ; Feng, Wei ; Zheng, Gan: Optimum Placement of UAV as Relays. In: *IEEE Communications Letters* 22 (2018), Nr. 2, S. 248–251. http://dx.doi.org/10.1109/LCOMM.2017.2776215. – DOI 10.1109/LCOMM.2017.2776215

[14] Community, F1TENTH Autonomous R.: *f1tenth_system: Drivers onboard f1tenth race cars*. https://github.com/f1tenth/f1tenth_system. – visited on 2024-02-09

[15] DOUGLAS, DAVID H. ; PEUCKER, THOMAS K.: ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10 (1973), Nr. 2, 112-122. http://dx.doi.org/10.3138/FM57-6770-U75U-7727. – DOI 10.3138/FM57–6770–U75U–7727

[16] In: Furrer, Fadri ; Burri, Michael ; Achtelik, Markus ; Siegwart, Roland: *RotorS – A Modular Gazebo MAV Simulator Framework*. Bd. 625. 2016. – ISBN 978–3–319–26054–9, S. 595–625

[17] Furtado, Joshua S. ; Liu, Hugh H. T. ; Lai, Gilbert ; Lacheray, Herve ; Desouza-Coelho, Jason: Comparative Analysis of OptiTrack Motion Capture Systems. (2019), S. 15–31. ISBN 978–3–030–17369–2

[18] FutureLearn, Univestity of O.: *How does an infrared motion capture system work? (Motion Capture: The Art of Studying Human Activity)*. https://www.futurelearn.com/info/courses/motion-capture-course/0/steps/272015. – visited on 2024-01-06

[19] Garrido-Jurado, S. ; Muñoz-Salinas, R. ; Madrid-Cuevas, F.J. ; Marín-Jiménez, M.J.: Automatic generation and detection of highly reliable fiducial markers under occlusion. In: *Pattern Recognition* 47 (2014), Nr. 6, 2280-2292. http://dx.doi.org/https://doi.org/10.1016/j.patcog.2014.01.005. – DOI https://doi.org/10.1016/j.patcog.2014.01.005. – ISSN 0031–3203

[20] Gautam, Alvika ; Sujit, P.B. ; Saripalli, Srikanth: Application of guidance laws to quadrotor landing. (2015), S. 372–379. http://dx.doi.org/10.1109/ICUAS.2015.7152312. – DOI 10.1109/ICUAS.2015.7152312

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[21] Giernacki, Wojciech ; Skwierczyński, Mateusz ; Witwicki, Wojciech ; Wroński, Paweł ; Kozierski, Piotr: Crazyflie 2.0 quadrotor as a platform for research and education in robotics and control engineering. (2017), S. 37–42. http://dx.doi.org/10.1109/MMAR.2017.8046794. – DOI 10.1109/MMAR.2017.8046794

[22] Glärbo: *Determine if a point is in a camera's field of view (3D).* https://math.stackexchange.com/questions/4144827/determine-if-a-point-is-in-a-cameras-field-of-view-3d. – visited on 2023-10-11

[23] In: Hönig, Wolfgang ; Ayanian, Nora: *Flying Multiple UAVs Using ROS*. Springer International Publishing, 2017. – ISBN 978–3–319–54927–9, 83–118

[24] Hönig, Wolfgang: *crazyflie_cpp: Standalone C++ library to use the Crazyflie quadrotor.* https://github.com/whoenig/crazyflie_cpp. – visited on 2024-02-09

[25] Jr, Roger R. L.: *FilterPy - Kalman filters and other optimal and non-optimal estimation filters in Python.* https://github.com/rlabbe/filterpy. – visited on 2024-03-03

[26] Julier, Simon J. ; Uhlmann, Jeffrey K.: New extension of the Kalman filter to nonlinear systems. In: *Defense, Security, and Sensing*, 1997

[27] Kalachev, Oleg: *Online ArUco markers generator.* https://github.com/okalachev/arucogen. Version: 2018

[28] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Journal of Basic Engineering* 82 (1960), 03, Nr. 1, 35-45. http://dx.doi.org/10.1115/1.3662552. – DOI 10.1115/1.3662552. – ISSN 0021–9223

[29] Kam, Hyeong R. ; Lee, Sung-Ho ; Park, Taejung ; Kim, Chang-Hun: RViz: A Toolkit for Real Domain Data Visualization. In: *Telecommun. Syst.* 60 (2015), oct, Nr. 2, 337–345. http://dx.doi.org/10.1007/s11235-015-0034-5. – DOI 10.1007/s11235–015–0034–5. – ISSN 1018–4864

[30] Koenig, N. ; Howard, A.: Design and use paradigms for Gazebo, an open-source multi-robot simulator. 3 (2004), S. 2149–2154 vol.3. http://dx.doi.org/10.1109/IROS.2004.1389727. – DOI 10.1109/IROS.2004.1389727

[31] Lange, Sven ; Sünderhauf, Niko ; Protzel, Peter: Autonomous Landing for a Multirotor UAV Using Vision. In: *In Workshop Proceedings of SIMPAR 2008 Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots* (2008), 01

[32] Li, Jianqiang ; Deng, Genqiang ; Luo, Chengwen ; Lin, Qiuzhen ; Yan, Qiao ; Ming, Zhong: A Hybrid Path Planning Method in Unmanned Air/Ground Vehicle (UAV/UGV) Cooperative Systems. In: *IEEE Transactions on Vehicular Technology*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

65 (2016), Nr. 12, S. 9585–9596. http://dx.doi.org/10.1109/TVT.2016.2623666. – DOI 10.1109/TVT.2016.2623666

[33] Lowe, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), Nov, Nr. 2, 91-110. http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94. – DOI 10.1023/B:VISI.0000029664.99615.94. – ISSN 1573–1405

[34] Macenski, Steven ; Foote, Tully ; Gerkey, Brian ; Lalancette, Chris ; Woodall, William: Robot Operating System 2: Design, architecture, and uses in the wild. In: *Science Robotics* 7 (2022), may, Nr. 66. http://dx.doi.org/10.1126/scirobotics.abm6074. – DOI 10.1126/scirobotics.abm6074

[35] Macsek, Markus ; Kastner, Wolfgang ; Bader, Markus: Mobile Robotik : EKF-SLAM mit optisch erkannten Markierungen zur Bestimmung einer Fahrzeugposition. (2016). http://dx.doi.org/https://doi.org/10.34726/hss.2016.36364. – DOI https://doi.org/10.34726/hss.2016.36364

[36] Marquardt, Donald W.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. In: *Journal of the Society for Industrial and Applied Mathematics* 11 (1963), Nr. 2, 431–441. http://www.jstor.org/stable/2098941. – ISSN 03684245

[37] Mueller, Mark W. ; Hamer, Michael ; D'Andrea, Raffaello: Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, S. 1730–1736

[38] Mueller, Mark W. ; Hehn, Markus ; D'Andrea, Raffaello: Covariance Correction Step for Kalman Filtering with an Attitude. In: *Journal of Guidance, Control, and Dynamics* 40 (2017), Nr. 9, 2301-2306. http://dx.doi.org/10.2514/1.G000848. – DOI 10.2514/1.G000848

[39] Nex, Francesco ; Remondino, Fabio: UAV for 3D mapping applications: a review. In: *Applied Geomatics* 6 (2014), Mar, Nr. 1, 1-15. http://dx.doi.org/10.1007/s12518-013-0120-x. – DOI 10.1007/s12518–013–0120–x. – ISSN 1866–928X

[40] O'Kelly, Matthew ; Sukhil, Varundev ; Abbas, Houssam ; Harkins, Jack ; Kao, Chris ; Pant, Yash V. ; Mangharam, Rahul ; Agarwal, Dipshil ; Behl, Madhur ; Burgio, Paolo ; Bertogna, Marko: F1/10: An Open-Source Autonomous Cyber-Physical Platform. (2019)

[41] Otsu, Nobuyuki: A Threshold Selection Method from Gray-Level Histograms. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1979), Nr. 1, S. 62–66. http://dx.doi.org/10.1109/TSMC.1979.4310076. – DOI 10.1109/TSMC.1979.4310076

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[42] Polvara, Riccardo ; Patacchiola, Massimiliano ; Sharma, Sanjay ; Wan, Jian ; Manning, Andrew ; Sutton, Robert ; Cangelosi, Angelo: Toward End-to-End Control for UAV Autonomous Landing via Deep Reinforcement Learning. (2018), S. 115–123. http://dx.doi.org/10.1109/ICUAS.2018.8453449. – DOI 10.1109/ICUAS.2018.8453449

[43] Preiss, James A. ; Honig, Wolfgang ; Sukhatme, Gaurav S. ; Ayanian, Nora: Crazyswarm: A large nano-quadcopter swarm. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, S. 3299–3304

[44] Quigley, Morgan ; Conley, Ken ; Gerkey, Brian ; Faust, Josh ; Foote, Tully ; Leibs, Jeremy ; Wheeler, Rob ; Ng, Andrew: ROS: an open-source Robot Operating System. In: *ICRA Workshop on Open Source Software* 3 (2009), 01

[45] Rodriguez-Ramos, Alejandro ; Sampedro, Carlos ; Bavle, Hriday ; Moreno, Ignacio G. ; Campoy, Pascual: A Deep Reinforcement Learning Technique for Vision-Based Autonomous Multirotor Landing on a Moving Platform. (2018), S. 1010–1017. http://dx.doi.org/10.1109/IROS.2018.8594472. – DOI 10.1109/IROS.2018.8594472

[46] Rucco, Alessandro ; Sujit, P. B. ; Aguiar, A. P. ; Sousa, João B. ; Pereira, F. L.: Optimal Rendezvous Trajectory for Unmanned Aerial-Ground Vehicles. In: *IEEE Transactions on Aerospace and Electronic Systems* 54 (2018), Nr. 2, S. 834–847. http://dx.doi.org/10.1109/TAES.2017.2767958. – DOI 10.1109/TAES.2017.2767958

[47] Silano, G. ; Aucone, E. ; Iannelli, L.: CrazyS: A Software-In-The-Loop Platform for the Crazyflie 2.0 Nano-Quadcopter. (2018), S. 352–357. http://dx.doi.org/10.1109/MED.2018.8442759. – DOI 10.1109/MED.2018.8442759. – ISSN 2473–3504

[48] Skoczylas, Marcin: Vision Analysis System for Autonomous Landing of Micro Drone. In: *Acta Mechanica et Automatica* 8 (2015), Nr. 4, 199–203. http://dx.doi.org/doi:10.2478/ama-2014-0036. – DOI doi:10.2478/ama–2014–0036

[49] Sowa, B.: *ROS wrapper for ArUco Opencv module*. https://github.com/fictionlab/ros_aruco_opencv. – visited on 2024-01-07

[50] Suzuki, Satoshi ; be, KeiichiA: Topological structural analysis of digitized binary images by border following. In: *Computer Vision, Graphics, and Image Processing* 30 (1985), Nr. 1, 32-46. http://dx.doi.org/https://doi.org/10.1016/0734-189X(85)90016-7. – DOI https://doi.org/10.1016/0734–189X(85)90016–7. – ISSN 0734–189X

[51] Thrun, S. ; Burgard, W. ; Fox, D.: *Probabilistic Robotics*. MIT Press, 2005 (Intelligent Robotics and Autonomous Agents series). https://books.google.at/books?id=k_yOQgAACAAJ. – ISBN 9780262201629

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[52] Traxxas: *Traxxas Slash 4x4 Premium Chassis*. https://traxxas.com/products/models/electric/6804Rslash4x4platinum. – visited on 2024-01-07

[53] Vedder, Benjamin: *VESC Project*. https://vesc-project.com/documentation. – visited on 2024-01-07

[54] Virtanen, Pauli ; Gommers, Ralf ; Oliphant, Travis E. ; Haberland, Matt ; Reddy, Tyler ; Cournapeau, David ; Burovski, Evgeni ; Peterson, Pearu ; Weckesser, Warren ; Bright, Jonathan ; van der Walt, Stéfan J. ; Brett, Matthew ; Wilson, Joshua ; Millman, K. J. ; Mayorov, Nikolay ; Nelson, Andrew R. J. ; Jones, Eric ; Kern, Robert ; Larson, Eric ; Carey, C J. ; Polat, İlhan ; Feng, Yu ; Moore, Eric W. ; VanderPlas, Jake ; Laxalde, Denis ; Perktold, Josef ; Cimrman, Robert ; Henriksen, Ian ; Quintero, E. A. ; Harris, Charles R. ; Archibald, Anne M. ; Ribeiro, Antônio H. ; Pedregosa, Fabian ; van Mulbregt, Paul ; SciPy 1.0 Contributors: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. In: *Nature Methods* 17 (2020), S. 261–272. http://dx.doi.org/10.1038/s41592-019-0686-2. – DOI 10.1038/s41592–019–0686–2

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.