RWTH Aachen University
Institut für Luft- und Raumfahrtsysteme

Diese Arbeit wurde vorgelegt am Institut für Luft- und Raumfahrtsysteme

von

Sean W. Hicken

Matrikel-Nr.: 439618

Master-Thesis

Entwicklung einer sensorgestützten Lösung zur gegenseitigen Lokalisierung und autonomen Annäherung zweier Objekte in einem Testbett für künftige Weltraumanwendungen

Aachen, 3. Februar 2025

Wissenschaftliche Leitung:	Univ.-Prof. Dr.-Ing. Eike Stumpf
Wissenschaftliche Betreuung:	Kim Goldschmidt, M.Sc. (RWTH)
Externe Betreuung:	Dipl.-Ing Falk Nohka (DLR)

1. Prüfer:	Univ.-Prof. Dr.-Ing. Eike Stumpf
2. Prüfer:	Dr.-Ing. Ralf Hörnschemeyer

Zusammenfassung

Autonomie stellt auf Grund der stetig wachsenden Relevanz von Raumfahrtprojekten in Wirtschaft und Forschung einen zentralen Systemaspekt in Raumfahrtmissionen dar. Einen besonderen Stellenwert nehmen hierbei Autonomous Rendezvous and Docking (AR&D) Funktionalitäten innerhalb aktueller und zukünftiger Missionsziele ein. Ziel dieser Abschlussarbeit ist es einen Überblick über sensorgestützte Lösungen zur Realisierung von AR&D-Funktionalitäten zu schaffen und die physikalischen Grundlagen dieser Sensorsysteme zu erläutern. Im Kontext der AR&D-Funktionalitäten wurde der Aspekt des autonomen Lokalisierens und navigierens Untersucht. Die Grundlage dieser Untersuchung stellen hierbei die Hauptphasen IV und V von Rendezvousmanöver dar. Zur Umsetzung der Untersuchung wurde nach einer differenzierten Auswahl von möglichen Sensorlösungen diese innerhalb eines Testbetts implementiert. Die softwaretechnische Verbindung der Sensorik mit der Lokalisierungs- und Navigationsalgorithmik wurde über das Softwareframework Robot Operating System 2 (ROS2) Umgesetzt. Zur Übertragbarkeit des Lokalisierungsproblems im dreidimensionalen Raum wurden die Freiheitsgrade (DOF's) des Testbetts von sechs auf zwei reduziert. Rahmenbedingung der Abbildbarkeit ist, dass das Testbett in der Lage sein muss innerhalb einer Büroumgebung autonom zu Navigieren sowie sich innerhalb dieser zu Lokalisieren. Die Untersuchung der Lokalisierungsfähigkeit im Raum mittels Simultaneous Localization and Mapping-Algorithmik (SLAM) wurde über eine statistische Auswertung realisiert. Hierbei wurden unter anderem die Kartendaten mit den realen Raumgrößen verglichen. Zusätzlich wurden die durch ROS2 generierten Koordinatentransformationen der realen Raumposition des Testbetts gegenübergestellt. Abschließend wurden die vom SLAM-Algorithmus erzeugten Kovarianzen der Raumpositionen über die Zeit analysiert. Bei der Untersuchung der Navigationsfähigkeiten wurde ein qualitatives Modell zur Analyse verwendet in welchem die Fähigkeit innerhalb eines vorher definierten Parkours zu navigieren betrachtet wurde. Hierbei wurden die benötigte Zeit, die Anzahl der Recoverys und ob der Parkour absolviert wurde analysiert. Die Ergebnisse dieser Untersuchung zeigten eine ausreichende Genauigkeit zur Durchführung von Nahbereichsmanövern wie von den Hauptphasen IV und V des Rendezvous und Dockings verlangt wird. Eine nähere Untersuchung der Übertragbarkeit dieser Fähigkeiten anhand der verwendeten Algorithmik auf sechs DOF's, sowie die Anwendbarkeit auf alternative Sensorlösungen bleibt Bestandteil zukünftiger Forschung.

Abstract

Autonomy represents a central system aspect in space missions due to it's increasingly growing relevance in space projects in both economy and research. Autonomous Rendezvous and Docking (AR&D) functionalities play a particularly important role within current and future mission objectives. The aim of this thesis is to provide an overview of sensor-based solutions for implementing AR&D functionalities and to explain the physical principles of these sensor systems. In the context of AR&D functionalities, the aspect of autonomous localization and navigation was investigated. The basis of this investigation were the main phases IV and V of rendezvous and docking maneuvers. For the implementation of the study, a differentiated selection of possible sensor solutions was made, which were then integrated within a testbed. The software-based connection of the sensors with the localization and navigation algorithms was realized using the software framework Robot Operating System 2 (ROS2). To make the localization problem transferable to three-dimensional space, the degrees of freedom (DOF's) of the testbed were reduced from six to two. The base condition for the system's capability was that the testbed should be able to navigate autonomously within an office environment and localize itself within it. The investigation of the localization capability using Simultaneous Localization and Mapping (SLAM) algorithms was carried out through statistical analysis. Among other things, the map data was compared with the real spatial dimensions of the office environment. Additionally, the coordinate transformations generated by ROS2 were compared with the ground truth of the testbed. Finally, the covariances of the spatial positions generated by the SLAM algorithm over time were analyzed. In examining the navigation capabilities, a qualitative model was used for analysis, which assessed the ability to navigate within a predefined course. This analysis included the time required, the number of recoveries, and whether the course was completed or not. The results of these analyses showed sufficient accuracy for performing proximity maneuvers, as required in the phases IV and V of rendezvous and docking. A more detailed investigation of the transferability of these capabilities based on the used algorithms to six DOFs, as well as the applicability to alternative sensor solutions, remains a subject for future research.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Allgemeine Symbole

$\vec{a}$	Beschleunigung	$[\ \frac{\mathrm{m}}{\mathrm{s}^2}\ ]$
a	Parameter des Satellite Clock Errors	$[\ \mathrm{s}\ ]$
α	Winkelbeschleunigung	$[\ \frac{\mathrm{rad}}{\mathrm{s}^2}\ ]$
$\angle$	Winkel	$[\ ^\circ\ ]$
b	Entfernung zwischen Kameras (baseline)	$[\ \mathrm{m}\ ]$
β	Geradenwinkel	$[\ ^\circ\ ]$
C	Kapazität	$[\ \mathrm{A\,h}\ ]$
c	Lichtgeschwindigkeit	$[\ \frac{\mathrm{m}}{\mathrm{s}}\ ]$
D	Dämpfungskoeffizient	$[\ \frac{\mathrm{N\,s}}{\mathrm{m}}\ ]$
d	Differentialoperator	$[\ -\ ]$
d	Distanz	$[\ \mathrm{m}\ ]$
$\Delta\delta$	Receiver Clock Offset	$[\ \mathrm{s}\ ]$
δ	Satellite Clock Error	$[\ \mathrm{s}\ ]$
Δ	Differenz	$[\ -\ ]$
$\vec{F}$	Kraft	$[\ \mathrm{N}\ ]$
f	Frequenz	$[\ 1/\mathrm{s}\ ]$
f	Brennweite	$[\ \mathrm{m}\ ]$
I	Intensität	$[\ \frac{\mathrm{W}}{\mathrm{m}^2}\ ]$
I	Strom	$[\ \mathrm{A}\ ]$

K	Federkonstante	[$\frac{\mathrm{N}}{\mathrm{m}}$]
L	Freiraumdämpfung	[dB]
l	Länge	[m]
λ	Wellenlänge	[m]
m	Masse	[kg]
n	Zählvariable	[–]
ω	Winkelgeschwindigkeit	[$\frac{\mathrm{rad}}{\mathrm{s}}$]
$\vec{p}$	Impuls	[N s]
P	Leistung	[W]
Φ	Phasenverschiebung	[°]
ϕ	Rollwinkel	[°]
ψ	Gierwinkel	[°]
$p(t)$	Signal über die Zeit	[dB]
q	Quaternion	[–]
R	Objektgröße	[m]
r	Radius	[m]
σ	Strahlungskoeffizienten	[–]
s	Strecke	[m]
sk	Skalierungsfaktor	[–]
t	Zeit	[s]
τ	Zeitverzug	[s]
Θ	Winkel zwischen Emitter und Geschwindigkeitsvektor	[°]
θ	Nickwinkel	[°]
$\mathbf{T}$	Transformationsmatrix	[**3x3**]
U	Spannung	[V]

u ..... Integrationskonstante [$-$]

v ..... Geschwindigkeit [$\frac{\mathrm{m}}{\mathrm{s}}$]

w ..... Winkelinkrement [$^\circ$]

x ..... x-Koordinate im kartesischen Koordinatensystem [$-$]

y ..... y-Koordinate im kartesischen Koordinatensystem [$-$]

Z ..... Encoderauflösung [$^\circ$/i]

z ..... z-Koordinate im kartesischen Koordinatensystem [$-$]

Tiefgestellte Indizes

0 ..... Initialgröße

A/B ..... Zuordnungssymbol bei zwei oder mehreren Objekten

ab ..... Absorbtion

B ..... Bias

D ..... Drift

Enc ..... Encoderzahnzahl

ext ..... extern

i ..... Zählvariable

k ..... Zählvariable

l ..... Links

P ..... Alterungsprozess

p ..... Zuordnung zu einem Punkt

R ..... Receiver

r ..... Rechts

r ..... Reflextion

T ..... Transmitter

t ..... Transmission

Hochgestellte Indizes

$\hat{}$	Annäherung an einen Wert
i	Zählvariable bei mehreren Zählvariablen pro Symbol

Abkürzungen

AOA	Angle of Arrival
API	Application Programming Interface
AR&D	Autonomous Rendezvous and Docking
AVGS	Advanced Video Guidance Sensor
BeAlMet	Beryllium-Alluminium Metal Matrix Composit
CAN	Controller Area Network
CCD	Charge Coupled Device
CEV	Crew Exploration Vehicle
CIO	Conventional International Origin
COTS	Commercial Orbit Transport System
CPU	Central Processing Unit
DDR5	Double Data Rate Synchronous 5
DDS	Data Distribution Service
DMIC	Digital Microphone
DOF	Degrees of Freedom
DSPK	Digital Speaker
DWA	Dynamic Window Approach
EM	elektromagnetisch
Enc1	Encoder 1
Enc2	Encoder 2
FDIR	Fault detection, Fault-Isolation and Recovery

FDM ..... Fused Deposition Modeling

FDOA ..... Frequency Difference of Arrival

FOV ..... Field of View

FPGA ..... Field Programmable Gate Array

FPS ..... Frames per Second

Gb ..... Giga Byte

GND ..... Erdung

GPIO ..... General Purpose Input Output

GPU ..... Graphics Processing Unit

HDD ..... Hard Disc Drive

HDMI ..... High Definition Multimedia Interface

HF ..... high frequency

I/O ..... Input/Output

I2c ..... Inter-integrated circuit

I2S ..... Inter-IC Sound

IC ..... Integrated Circuit

IF ..... intermediate frequency

IGRF ..... International Geomagnetic Reference Field

IGS ..... International GNSS Service

ILR ..... Institute of Aerospace Systems

IN1 ..... Eingang 1

IN2 ..... Eingang 2

LF ..... low frequency

LiPo ..... Lithium Polymere

MIPI ..... Mobile Industry Processor Interface

NASA	National Aeronautics and Space Administration
NGAVGS	Next Generation Advanced Video Guidance Sensor
PDOA	Phase Difference of Arrival
PLA	Polylactide
RAM	Random-Access Memory
RFID	Radiofrequenzidentifikation
RF	Radio frequency
ROS2	Robot Operating System 2
RPOD	Rendezvous, Proximity Operations and Docking
SD	Secure Digital
SHF	super high frequency
SLAM	Simultaneous Localization and Mapping
SPI	Serial Peripheral Interface
TDOA	Time Difference of Arrival
UART	Universal Asynchronous Receiver Transmitter
UHF	ultra high frequency
USB	Universal Serial Bus
XML	Extensible Markup Language

Danksagung

Ich würde gerne an dieser Stelle dem DLR-Bremen sowie den Menschen am ILR danken die durch ihre tatkräftige Unterstützung diese Masterarbeit ermöglicht haben.

Dieser Dank gilt insbesondere meinem Betreuer bei DLR Falk Nohka, jedoch auch meinen Unterstützern aus privatem Umkreis, hierzu zählen vor allem und ohne deren Unterstützung diese Arbeit nicht möglich gewesen wäre.

Vielen dank dass ihr dabei wart!

Sean W. Hicken

1 Einleitung

Die Lokalisierung im Nah und Fernbereich ist ein zentrales Problem für Rendezvousmanöver in der Raumfahrt. Einer der Hauptaufgaben hierbei besteht darin, geeignete Sensorik zu implementieren welche es dem Raumfahrtsystem ermöglicht einerseits seine Umgebung wahrzunehmen und andererseits aus diesen Informationen die Position im Raum, sowie relativ zu anderen Objekten zu ermitteln.

Dieses für Rendezvousmanöver in der Raumfahrt im allgemeinen bestehende Problem erhält besondere Relevanz im Kontext von Systems of Systems, Satelliten-Cluster, Wartungs-, Explorations-Missionen und autonomen Dockingmanövern. Ziel dieser Abschlussarbeit ist es, verschiedene physikalische Methoden zur Lösung des Lokalisierungsproblems gegenüber zu stellen. Zusätzlich soll aus diesen Erkenntnissen geeignete Sensorik erschlossen werden und diese im Kontext der zu bewältigenden Probleme validiert und ihre Grenzen, sowie die Vor- und Nachteile erarbeitet werden.

Hierbei soll sich primär auf die Generierung und Verwendung von Sensordaten zur gegenseitigen Lokalisierung und autonomen Annäherung zweier Objekte fokussiert werden. Zusätzlich wird ebenfalls die Abbildbarkeit und Implementierung der Sensorik in einem Testbett berücksichtigt.

Rendezvousmanöver lassen sich in fünf Hauptphasen unterteilen (siehe [7] und [44]). Diese Phasen bestehen aus der I Launch-Phase, II Orbit-Injection-Phase, III Orbit-Phasing-Phase, IV Far-Rendezvous-Operation-Phase und V Close-Rendezvous-Operation-and-Spacecraft-Mating-Phase. Für Phase I-IV sind Lokalisierungen im Fernbereich neben der vor dem Launch geplanten Trajektorie von zentraler Bedeutung und dienen der Regelung und Anpassung der Bahnenelemente [7]. In Phase IV-V sind neben den Informationen welche aus der Lokalisierung im Fernbereich generiert werden (Beispielsweise durch GNSS-Dienste), auch Informationen aus Lokalisierungsmethoden im Nahbereich von zentraler Bedeutung. Grund hierfür sind die beschränkte Auflösung der Lokalisierungsmethoden im Fernbereich wie beispielsweise GPS und Bodenradar [44]. Für kommerzielle Raumfahrtysteme bestehen bereits Ansätze für implementierte Sensorlösungen zur Lokalisierung im Nah und Fernbereich [14]. Dabei stellen die durch die NASA und ESA in der Vergangenheit verwendeten Systeme, wie das RVS3000 (siehe Ref. [19]) und das von Howard u. a. [14] gezeigte System, geeignete Produkte dar, um Daten mittels Sensorik zu generieren und eine Lokalisierung sowie die Bestimmung relativer Positionen zweier Objekte zueinander umzusetzten.

Diese Informationen und Sensordaten können zu dem genutzt werden um weitere Funktionalitäten für Rendezvousmanöver zu ermöglichen und den Umfang der Missionsanalysedaten zu erweitern.

Zu diesen Funktionalitäten gehört neben der allgemeinen Trajektorieplanung wie von Duch [7] erwähnt, auch das Autonomous Rendezvous and Docking (AR&D) welches eine zentrale Rolle in der heutigen und zukünftigen Raumfahrt spielt. Ruel u. a. [44] verweist hierzu in seinem Paper auf die 2010 erfolgte Klassifizierung der NASA von AR&D Technologien als "...key enabling technology. This technology is required not only for future space exploration initiatives but also to develop the ability to extend the life of valuable assets in orbit via in-orbit servicing."[44, S. 1]. Beispiele für die Umsetzung von AR&D Systemen mit kooperativen Zielsystemen stellen das ATV der ESA und das japanische HTV dar [44]. AR&D stellt unter anderem wegen der rapide wachsenden Zahl an Weltraumsystemen und der daraus resultierenden Anforderung an Autonomie in Rendezvous- und Docking-Manövern, eine zentrale Rolle für zukünftige Weltraumexplorationen dar [30]. Weitere Anwendungsbeispiele für Missionen in welchen AR&D eine zentrale Rolle spielt sind „in Space Assembley“ und Wartungsmissionen [30]. Docking und Rendezvous Vorgänge vieler Raumfahrtsysteme, wie Beispielsweise der der ISS, sind heute immer noch stark von manuellen Steuerungen abhängig um sichere Docking- und Rendezvous-Manöver zu gewährleisten. Aus diesem Grund stellt AR&D und die sensorbedingte Verifizierung ihrer Anwendbarkeit eine übergeordnete Rollte in der aktuellen Forschung dar. Hierbei stellt sich vor allem im Bereich von Satellitenclustern und Systems of Systems die Frage nach einer zugänglichen und volumeneffizienten Möglichkeit Kopplungsprozesse, dass erkennen von kooperativen und nicht kooperativen Zielen, sowie das Mapping der Umgebung innerhalb eines Satellitenclusters zu ermöglichen.

Die Qualität der Lösung dieser Aufgaben ist vor allem im Bereich Autonomie stark von den Daten welche von der Onboard Sensorik generiert werden abhängig. Die Evaluierung dieser Daten sowie die daraus resultierenden Funktionalität spielt eine zentrale zu lösende Aufgabe für zukünftige Raumfahrtsysteme und soll in dieser Abschlussarbeit untersucht werden.

2 Physikalische Grundlagen und Sensortechnologien

Das folgende Kapitel dient der Erläuterung und Definition der in dieser Arbeit verwendeten physikalischen Grundlagen. Diese fungieren als Fundament der Messmethoden, welche zur Positions- und Orientierung-bestimmung genutzt werden. Des Weiteren werden in diesem Kapitel Sensortechnologien vorgestellt, welche auf den vorher benannten physikalischen Grundlagen und Methoden basieren. Grund hierfür ist einen Überblick über die Funktionsweise von Lage- und Positionssensorik zu geben sowie die Basis einer Auswahl für die im Testbett zu implementierende Sensorik zu bieten.

2.1 Physikalische Grundlagen

In diesem Unterkapitel werden die physikalischen Grundlagen dargestellt, welche genutzt werden können, um Informationen über die Position und Ausrichtung eines Körpers, sowie die daraus entstehende relative Position und Ausrichtung zweier Körper zueinander zu ermitteln. Die mathematisch-physikalische Grundlage zur Beschreibung von Position und Lage eines beliebigen Körpers lässt sich im dreidimensionalen Raum auf die translatorischen und rotatorischen Freiheitsgrade und der Ermittlung dieser reduzieren. Diese Freiheitsgrade sind ebenfalls unter dem Akronym DOF (degrees of freedom) bekannt. Diese Freiheitsgrade beschreiben die absolute Position in x-,y-,z-Richtung, sowie die Eulerwinkel ϕ, θ, ψ (Roll-, Nick- und Gier-Winkel) des Körpers im Bezug zu einem Referenzsystem. Die folgenden physikalischen Methoden stellen unterschiedliche Möglichkeiten dar, diese Größen direkt und oder indirekt zu bestimmen.

2.1.1 Time Difference of Arrival

Die Methode „Time Difference of Arrival“ (TDOA) stellt ein Wegzeitverfahren dar mit der die absolute Entfernung zweier Objekte zueinander bestimmt werden kann. Grundlage der TDOA sind die Bewegungsgleichungen, welche durch das zweite newton'sche Gesetz (siehe Gleichung 2.1) hergeleitet werden können [8]

$$\frac{\mathrm{d}\vec{p}}{\mathrm{d}t} = m\,\vec{a} = \sum\nolimits_i \vec{F}_i. \tag{2.1}$$

Unter der Annahme der Impulserhaltung ergeben sich folgende Ausdrücke

$$\frac{\mathrm{d}\vec{p}}{\mathrm{d}t} = m\,\vec{a} = \sum\nolimits_i \vec{F_i} = 0, \tag{2.2}$$

$$0 = \int m\,\vec{a}\,\mathrm{d}t = m\,a(t)\,t + u, \tag{2.3}$$

beziehungsweise

$$m\,a(t)\,t + u = m\,v(t)\;+\;v_0, \tag{2.4}$$

wobei die Integrationskonstante u die Anfangsgeschwindigkeit darstellt. Nach erneuter Integration über die Zeit ergeben sich die Zusammenhänge

$$\int m\,a(t)\,t + u\,\mathrm{d}t = \int m\,v(t)\;+v_0\,\mathrm{d}t \tag{2.5}$$

und

$$m\,a(t)\,t^2 + u\,t = m\,v(t)\,t + v_0\,t = m\,s(t) + s_0. \tag{2.6}$$

Im Kontext der Positionsermittlung in der Raumfahrt werden vorwiegend elektromagnetische Wellen für TDOA-Verfahren verwendet. Daher gilt in Gleichung 2.6 für die Masse m=0. Des Weiteren ist die Geschwindigkeit durch die Lichtgeschwindigkeit c gegeben. Da diese eine konstante, nicht beschleunigte Größe ohne Initialgeschwindigkeit darstellt, vereinfacht sich die Gleichung zu [1, S.161]

$$c\,t = s(t)\;+\;s_z. \tag{2.7}$$

Die Ergänzung des Terms s_z in Gleichung (2.7) stellt eine Störgröße in der Berechnung der Bewegung dar. Diese Störgröße kann, je nach Anwendungsbereich dieses Verfahrens, an verschiedenen Punkten im Übertragungsweg entstehen. Diese Störgröße äußert sich in längeren Wegzeiten und daraus resultierenden Abweichungen der Strecke in der Berechnung. Da diese Störgröße stark vom Anwendungsfall abhängig sind, wird diese näher im Unterkapitel 2.2 behandelt.

Die TDOA-Methode beruht auf Laufzeitdifferenzen im Sendeweg eines Signals [1, S.164]. Bei dieser Methode wird beispielsweise ein zeitcodiertes RF-Signal eines Transmitters an einen Receiver gesendet. An Transmitter und Receiver werden für diese Methode, vor allem bei langen Übertragungswegen, hohe Anforderungen an die Präzision und Synchronität der Zeitmessungen

gestellt[49, S.4]. Nach Eintreffen des zeitcodierten RF-Signals wird die Zeitdifferenz zwischen Senden und Empfangen ermittelt und über die Gleichung (2.7) die korrelierende Distanz bestimmt. Im Allgemeinen muss bei Verwendung von elektromagnetischen Wellen in zwei Störgrößen unterschieden werden. Dem Pfadverlust und äußere Störgrößen, welche beispielsweise durch Schwankungen der Zustandsgrößen des Transmissionsmediums ausgelöst werden können. Bei geringeren Abständen oder dem fehlen eines Transmissionsmediums, kann die äußere Störgröße s_z vernachlässigt werden [1, S.161].

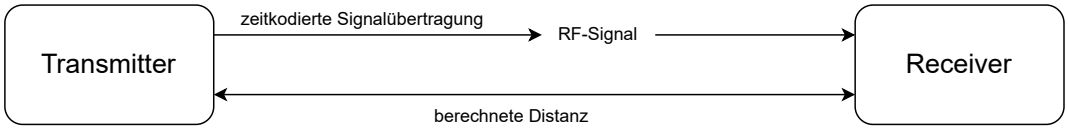


Abbildung 2.1: Schaubild TDOA

Pfadverluste im Übertragungsweg einer Satellitenkommunikation mittels RF-Signalen lassen sich in drei Untergruppen aufteilen: „Interner Kabelverlust“, „Freiraumdämpfung“ und „Übertragungsverluste“. Der „interne Kabelverlust“ beträgt im Durchschnitt 10% der Sendeleistung [35, S.41], der „Übertragungsverlust“ ist abhängig von der atmosphärischen Störung und somit abhängig von Ort und Höhe des Receivers und Transmitters. Die „Freiraumdämpfung“ ergibt sich zu [35, S.41]

$$L = (\frac{4\pi r}{\lambda})^2. \tag{2.8}$$

Wie mittels der TDOA-Methode eine genaue Lokalisierung realisiert werden kann, ist dem Kapitel 2.1.4 zu entnehmen.

2.1.2 Frequnecy Difference of Arrival

Die FDOA-Methode (Frequency Difference of Arrival) stellt eine weitere Möglichkeit dar, um eine Emitterlokalisierung zu realisieren. Hierbei werden zwei sich bewegende Receiver genutzt, um mittels des Dopplereffekts eine Frequenzverschiebung im empfangenen Signal zu messen, siehe Abbildung 2.2. Diese wird genutzt um den Winkel zwischen Emitter und Receiver zu bestimmen [1, S.167]. Die gemessene Frequenz des Receivers$_1$ ist hierbei abhängig von der Transmitterfrequenz, der Geschwindigkeit des Receivers und dem Winkel zum Emitter und gegeben durch den Zusammenhang[1, S.167]

$$f_R = f_T\{1 + [\frac{v_R \cos(\Theta)}{c}]\}. \tag{2.9}$$

f_R stellt die gemessene Receiverfrequenz, f_T die Transmitterfreqeuenz, v_R die Geschwindigkeit des Receivers, Θ den Winkel zwischen Geschwindigkeitsvektor und Transmitter und c die Lichtgeschwindigkeit dar.

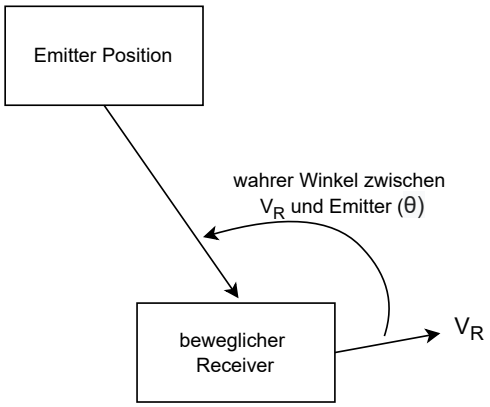


Abbildung 2.2: FDOA mit einem Receiver

Werden zwei Receiver mit bekannter Entfernung zueinander und jeweiligen Geschwindigkeitsvektoren betrachtet (siehe Abbildung 2.3), kann eine Kurve ermittelt werden, welche die möglichen Positionen des Transmitters darstellt. Diese Kurve wird auch „Iso-Frequency-Curve“ genannt. Der mathematische Zusammenhang hierfür ergibt sich zu [1, S.169]

$$\Delta f = f_T\{1 + [\frac{v_{R2}\cos(\Theta_2) - v_{R1}\cos(\Theta_1)}{c}]\}. \tag{2.10}$$

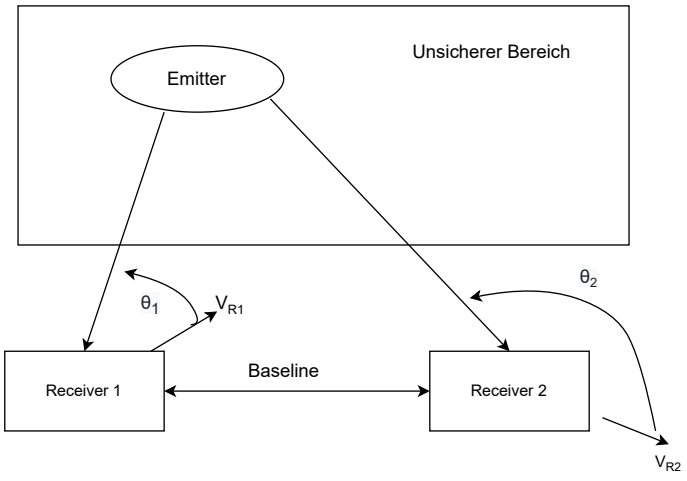


Abbildung 2.3: FDOA mit zwei Receiver

Die Indizierung der Variablen der Abbildung 2.3 folgt dem Schema analog zur Abbildung 2.2. Ähnlich der Trilateration (siehe Kapitel 2.1.4), kann mittels eines weiteren Receivers die genaue Position des Transmitters auf der Kurve ermittelt werden [1, S.169]. Die FDOA-Methode kann ebenfalls mit der TDOA-Methode kombiniert werden, um die Anzahl der Datenpunkte, sowie die Auflösung des Messverfahrens zu verbessern [1, S.169].

2.1.3 Phase Difference of Arival

Die „Phase Difference of Arival“-Methode (PDOA) ermöglicht es, empfangene elektromagnetische Signale zu interpretieren, um aus diesen Erkenntnisse über die Richtung und Entfernung der Emissionsquelle zu gewinnen. Die PDOA-Methode kann außerdem verwendet werden, um den „Angle of Arrival“ (AOA) des Signals zu bestimmen. Die AOA-Methode stellt eine Kombination aus der PDOA- und TDOA-Methode dar. Für die PDOA-Methode sind mindestens 2 Receiverantennen sowie eine Emissionsquelle notwendig. Die folgende Herleitung der mathematischen Zusammenhänge bezieht sich auf den in Abbildung 2.4 beschrieben Aufbau.

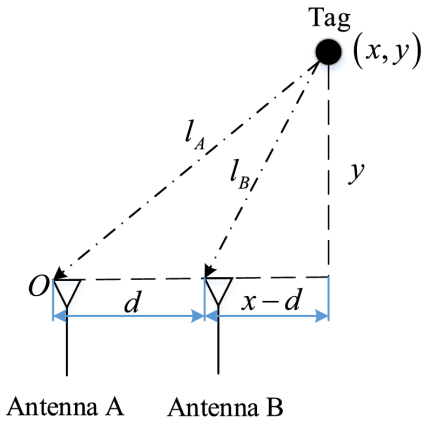


Abbildung 2.4: Aufbau des für die Herleitung verwendeten Modells [5]

.

Unter der Annahme, dass die Distanz zwischen den Antennen $d = \lambda/2$ entspricht, kann mittels der TODA-Methode die Entfernung zwischen Transmitter und Antenne A (l_A) bestimmt werden [53, S.3]. Das Verhältnis der Längen l_A und l_B kann als

$$l_B = l_A - \Delta l \tag{2.11}$$

ausgedrückt werden. Wenn Antenne B als Referenzpunkt gewählt wird, kann das empfangene Signal an Antenne A dem vom Emitter gesendeten Signal *p(t)* gleichgesetzt werden. Somit ergeben sich die Formulierungen [53, S.3]

$$r_A(t) = p(t) \tag{2.12}$$

und

$$r_B(t) = p(t)e^{-j\Phi} = p(t)e^{-j2\pi f\tau} \tag{2.13}$$

wobei Φ die Phasenverschiebung und f die Frequenz des übertragenen Signals darstellt. τ bildet hierbei den korrespondierenden Zeitverzug ab [53, S.1]. Eine Annäherung der Phasenverschiebung, kann somit wie folgt ausgedrückt werden

$$\hat{\Phi} = \angle \frac{r_B(t)}{r_A(t)}. \tag{2.14}$$

Nach Umstellen der Gleichung 2.14 ergibt sich für den Zeitverzug

$$\hat{\tau} = \frac{\hat{\Phi}}{2\pi f}. \tag{2.15}$$

Hieraus folgt für den Unterschied in der Weglänge der beiden Antennen

$$\Delta = \hat{\tau}c, \tag{2.16}$$

wobei c die Lichtgeschwindigkeit darstellt [53, S.3]. Im Folgenden kann nun mittels des Kosinus der AOA zwischen Transmitter und Antenne A wie folgt berechnet werden (siehe Abbildung 2.4)

$$\cos\Phi_A = \frac{x}{l_A} = \frac{l_A^2 + d^2 - (l_A - \Delta\hat{l})^2}{2l_A d}. \tag{2.17}$$

Durch Umformulierung der Gleichung 2.17 ergibt sich für die x-Koordinate

$$x = \frac{d^2 + 2l_A\Delta\hat{l} - 2\Delta\hat{l}}{2d} = (l_A - \frac{\Delta\hat{l}}{2})\frac{\Delta\hat{l}}{d} + \frac{d}{2}. \tag{2.18}$$

Mit Hilfe des Satzes des Pythagoras ergibt sich für die y-Koordinate die Gleichung

$$y = \pm\sqrt{l_A^2 - x^2}. \tag{2.19}$$

Nach einsetzen von Gleichung 2.19 in Gleichung 2.18 kann die y-Koordinate ebenfalls durch

$$y = \pm \frac{\sqrt{(1 - (\frac{\Delta\hat{l}}{d})^2)(4l_A^2 - 4l_A\Delta\hat{l} + \Delta\hat{l}^2 - d^2)}}{2} \tag{2.20}$$

ausgedrückt werden. Da der Abstand der Antennen d mathematisch sehr viel kleiner als der Abstand zwischen Antenne und Emitter A (l_A) ist, lässt sich schlussfolgern, dass der Wert für d^2 viel kleiner als die Größe l_A^2 ist. Daher kann die y-Koordinate näherungsweise zu [53, S.4]

$$y \approx \pm(l_A - \frac{\Delta\hat{l}}{2})\sqrt{1 - (\frac{\Delta\hat{l}}{d})^2}, \tag{2.21}$$

berechnet werden. Mit den Gleichungen 2.21 und 2.18 ergibt sich somit ein Ausdruck um die Koordinaten der Position eines Punktes P im zweidimensionalen-Raum zu beschreiben [53, S.4].

Abbildung 2.5 zeigt eine Erweiterung des Aufbaus, mit welchem mittels der PDOA Methode durch hinzufügen eines weiteren Sensors die Position eines Punktes im dreidimensionalen-Raum ermittelt werden kann. Die trigonometrischen Zusammenhänge hierfür folgen analog zu den vorherigen Herleitungen.

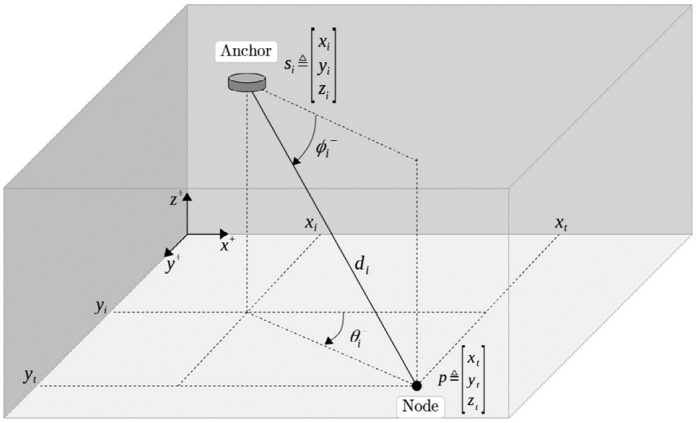


Abbildung 2.5: PDOA, Aufbau im 3D-Raum [5]

.

2.1.4 Trilateration

Die Trilateration ist ein zentraler Aspekt von Positionsbestimmungsmethoden. In dieser Methode wird mittels der Distanzen zwischen dem zu bestimmenden Objekt und bekannten statio-

nären oder beweglichen Objekten die genaue Position des zu bestimmenden Objektes im Raum ermittelt [37, S.9].

Die Positionsbestimmung durch Trilateration basiert auf der Konzept der Ermittlung der Schnittpunkte von Kreisgeometrien. Hierbei werden über die TDOA-, FDOA-, oder PDOA-Methode die Entfernung zwischen Transmitter und Receiver ermittelt. Die möglichen Positionen, an welchen sich der Receiver befinden kann, wird hierbei durch einen Kreis beschrieben, dessen Radius die gemessene Distanz zu dem Transmitter darstellt.

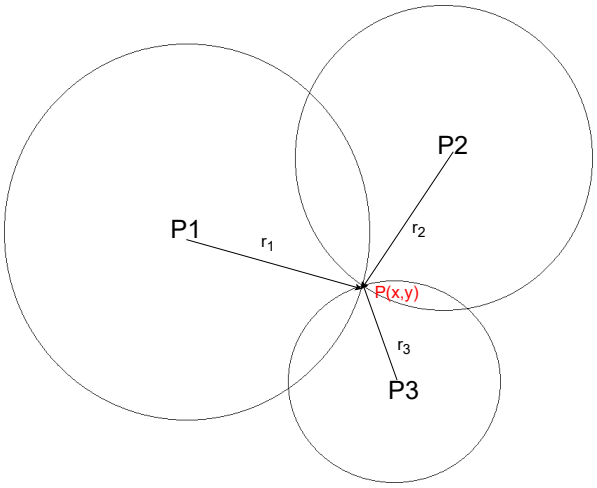


Abbildung 2.6: Trilateration im zweidimensionalen Raum: P1-P3 stellen die Transmitter dar, r_1-r_3 die Radien der Transmitter und P(x,y) den Receiver

Wird dieses Verfahren nun für mehrere Transmitter wiederholt, ist es möglich mittels drei Transmittern, deren Position im Raum bekannt sind, die genaue Position des Receivers, im zweidimensionalen Raum zu bestimmen (siehe Abbildung 2.6). Das grundlegende Verfahren hierzu wird im Folgenden erläutert.

Zur Darstellung eines Punktes auf einem Kreis, kann die Kreisgleichung in kartesischen Koordinaten herangezogen werden [37, S.9]

$$r_i^2 = (x_i - x_p)^2 + (y_i - y_p)^2. \tag{2.22}$$

Diese Gleichung kann nach Anwendung der binomischen Formeln zu

$$r_i^2 = x_i^2 + x_p^2 - 2x_i x_p + y_i^2 + y_p^2 - 2y_i y_p \tag{2.23}$$

umgeschrieben werden.

Unter Berücksichtigung von k Transmittern ergibt sich die Schreibweise

$$r_k^2 = x_k^2 + x_p^2 - 2x_k x_p + y_k^2 + y_p^2 - 2y_k y_p. \tag{2.24}$$

Gleichung (2.24) wird von Gleichung (2.22) subtrahiert, um eine passende Umformung für eine Matrixschreibweise zu generieren [37, S.9]

$$r_i^2 - r_k^2 + x_k^2 + y_k^2 - x_i^2 - y_i^2 = 2(x_k - x_i)x_p + 2(y_k - y_i)y_p. \tag{2.25}$$

Im in Abbildung 2.7 dargestellten Beispiel werden 4 Transmitter verwendet, um die Position des Receivers zu bestimmen. Im zweidimensionalen Raum wären nur drei Transmitter notwendig, um das Gleichungssystem 2.26 vollständig zu bestimmen. Der vierte Transmitter soll in diesem Fall das Vorgehen mit mehreren Transmittern verdeutlichen [37].

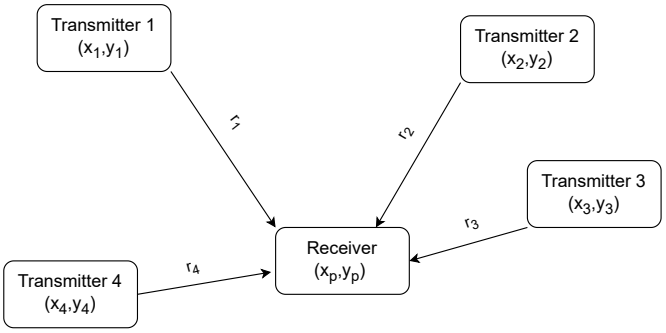


Abbildung 2.7: Trilateration im 2-dimensionalen Raum mit 4 Receivern

$$\begin{bmatrix} 2(x_2 - x_1) & 2(y_2 - y_1) \\ 2(x_3 - x_1) & 2(y_3 - y_1) \\ 2(x_4 - x_1) & 2(y_4 - y_1) \end{bmatrix} \begin{bmatrix} x_p \\ y_p \end{bmatrix} = \begin{bmatrix} r_1^2 - r_2^2 + x_2^2 + y_2^2 - x_1^2 - y_1^2 \\ r_1^2 - r_3^2 + x_3^2 + y_3^2 - x_1^2 - y_1^2 \\ r_1^2 - r_4^2 + x_4^2 + y_4^2 - x_1^2 - y_1^2 \end{bmatrix} \tag{2.26}$$

Das lineare Gleichungssystem 2.26 mit den Unbekannten x_p und y_p ist in dieser Form eindeutig lösbar.

Zur Positionsbestimmung mittels Trilateration im dreidimensionalen Raum ist es notwendig, an Stelle der Kreisgleichungen Kugelgleichungen zu implementieren sowie den Parameter der z-Komponente hinzuzufügen. Das Vorgehen hierbei erfolgt analog zu dem für den zweidimensiona-

len Raum. Hieraus folgt das Gleichungssystem 2.7 zur Bestimmung der Position eines Receivers im dreidimensionalen Raum.

$$\begin{bmatrix} 2(x_2-x_1) & 2(y_2-y_1) & 2(z_2-z_1) \\ 2(x_3-x_1) & 2(y_3-y_1) & 2(z_3-z_1) \\ 2(x_4-x_1) & 2(y_4-y_1) & 2(z_4-z_1) \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} r_1^2-r_2^2+x_2^2+y_2^2+z_2^2-x_1^2-y_1^2-z_1^2 \\ r_1^2-r_3^2+x_3^2+y_3^2+z_3^2-x_1^2-y_1^2-z_1^2 \\ r_1^2-r_4^2+x_4^2+y_4^2+z_4^2-x_1^2-y_1^2-z_1^2 \end{bmatrix} \quad (2.27)$$

Die Größen x_p, y_p und z_p stellen die Unbekannten der zu ermittelnden Position des Receivers dar. Die Größen x_i, y_i und z_i stellen die bekannten Positionen der Transmitter dar. Das Gleichungssystem weist drei unabhängige Gleichungen und drei Unbekannte auf und ist somit vollständig bestimmbar.

2.1.5 Triangulation

Durch die Triangulation kann durch Winkelbeziehungen und Abstände die Positionsberechnung eines Punktes im Raum realisiert werden. Diese Methode kommt vor allem in Stereokameras und bei geodätischen Messungen zum Einsatz, um Tiefeninformationen aus zwei Aufnahmen zu generieren. Im Folgenden werden die physikalischen Grundlagen zur Triangulation am Beispiel einer Stereokamera erläutert.

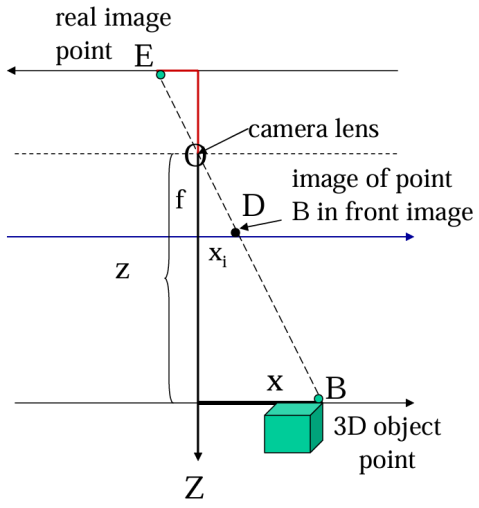


Abbildung 2.8: Triangulation mit einer Kamera über Strahlensätze [39, S.11]

Betrachtet man das in Abbildung 2.8 dargestellte Beispiel mit einer Kamera, sind die dargestellten Entfernungen über den Strahlensatz bestimmbar [39, S.5].

Das Verhältnis der Entfernungen beträgt demnach [39, S.11]

$$\frac{x_i}{f} = \frac{x}{z}. \tag{2.28}$$

Die Größe f stellt hierbei die Brennweite der Kamera dar. Diese Zusammenhänge können nun auf zwei optische Systeme mit parallelen optischen Achsen und bekanntem Abstand zueinander angewendet werden. Der Aufbau hiervon folgt analog zu Abbildung 2.9 [39, S.12].

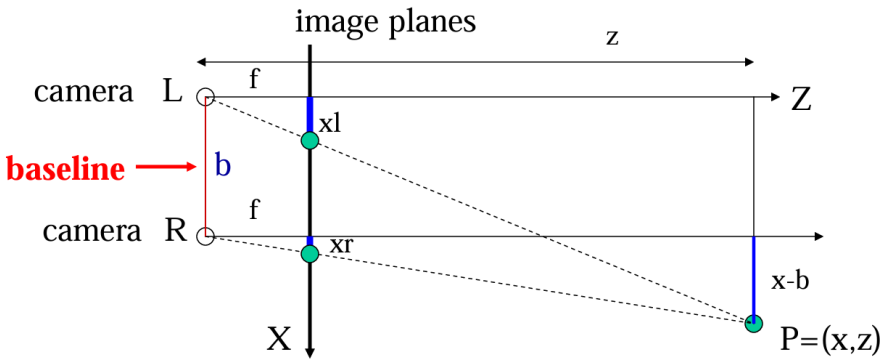


Abbildung 2.9: Triangulation mit einer Stereokamera über trigonometrische Zusammenhänge [39, S.29]

Wobei sich die Abstände analog zur Abbildung (2.28) zu

$$\frac{z}{f} = \frac{x}{x_l}, \tag{2.29}$$

$$\frac{z}{f} = \frac{x-b}{x_r}, \tag{2.30}$$

und

$$\frac{z}{f} = \frac{y}{y_l} = \frac{y}{y_r} \tag{2.31}$$

ergeben [39, S.12]. Die Größen y_r und y_l stellen in Gleichung 2.31, die korrespondierenden Größen der y-Achse dar. Diese Größen stellen zusammen mit x_r und x_l die Bildpunktkoordinaten des zu ermittelnden Punktes dar [39, S.11]. Daraus folgt, für die Entfernungsbestimmung eines Bildpunktes mittels Triangulation via Stereokamera (siehe Abbildung 2.9) für die z-Koordinate folgender Zusammenhang

$$z = \frac{f\, b}{x_l - x_r}. \tag{2.32}$$

Analog gilt für die x- und y-Koordinate

$$x = x_l \frac{z}{f} = b + x_r \frac{z}{f} \tag{2.33}$$

und

$$y = y_l \frac{z}{f} = y_r \frac{z}{f}. \tag{2.34}$$

Voraussetzung für die Ermittlung der genauen Raumkoordinaten ist, dass die Brennweite sowie die Entfernung der Kameras zu einander bekannt sind. Dies erfordert im Realfall eine Kalibrierung des Systems. Kalibrierungen nehmen einen übergeordneten Stellenwert in der Kameratechnik ein [39, S.14] und werden im Kapitel 2.2.3 behandelt.

2.1.6 Accelerometrie

Ziel der Accelerometrie ist die Messung der Beschleunigung a eines Punktes. Die Beschleunigung a wird durch die erste Ableitung der Geschwindigkeit nach der Zeit beschrieben [11, S.371]

$$a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2}. \tag{2.35}$$

Grundlegend basieren alle Beschleunigungsmessungen auf dem zweiten Newton'schen Gesetz [11, S.371]

$$\mathbf{F} = m\,\mathbf{a} \tag{2.36}$$

wobei F die Kraft, m die Masse und a die Beschleunigung darstellt. Ein Modell zur Beschreibung von beschleunigten Körpern ist der Federdämpfer (siehe Abbildung 2.10, dessen Bewegung über die Differentialgleichung

$$F_{\text{ext}} = m\frac{\mathrm{d}x^2}{\mathrm{d}^2t} + D\frac{\mathrm{d}x}{\mathrm{d}t} + K\ x \tag{2.37}$$

beschrieben werden kann [11, S.371].

D steht hierbei für den Dämpfungskoeffizienten, K für die Federkonstante, F_{ext} für die von außen wirkende Kraft und x für den Weg der Verschiebung [11, S. 371].

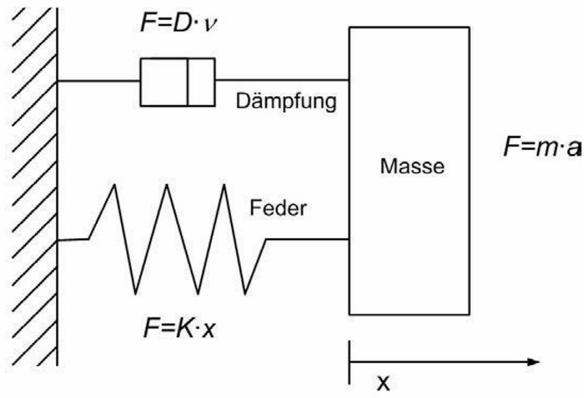


Abbildung 2.10: Schaubild Federdämpfer [11, S. 372]

.

Analog für die Winkelbeschleunigung

$$\alpha = \frac{\mathrm{d}\omega}{dt} \tag{2.38}$$

gilt.

In technischen Anwendungen wird die Wegmessung über die Zeit häufig durch DMS-Systemen realisiert. Aus diesen Daten kann nach der Messung durch Integration der Wegänderung über die Zeit die Beschleunigung auf den Massenschwerpunkt berechnet werden. Siehe Kapitel 2.2.8 zur Veranschaulichung verschiedener DMS-Systeme und ihrer technischen Integration.

2.2 Sensortechnologien

Sensoren können in zwei Hauptkomponenten unterteilt werden, dem Sensorelement und der Auswerteelektronik [11, S.1]. Die zu messenden, nicht elektrische Eingangsgrößen werden im Sensorelement in ein elektrisches Ausgangssignal gewandelt [11, S.1]. Sensoren stellen die Grundlage der Interaktion von technischen Anwendungen mit der physischen Welt dar und sind somit Bestandteil jeder technischen Anwendung, die mit ihrer Umgebung interagieren soll. In diesem Kapitel erfolgt eine Aufstellung gängigster Sensortechnologien zur Lokalisierung von Objekten im Nah- und Fernbereich.

2.2.1 Light Imaging Detection and Ranging

Light Imaging Detection and Ranging-Systeme (LIDAR-Systeme) (siehe Abbildung 2.12b) sind optische Messsysteme die auf dem Prinzip der TDOF basieren, um Abstände innerhalb einer Ebene oder bei 3D-LIDAR-Systemen innerhalb eines gewissen Bereichs, zu ermitteln [11, S.284]. Die Messbereiche sind in der Regel durch vom Hersteller angegebenen Winkelgrößen und Distanzen definiert und von dem inneren Aufbau des LIDAR-Systems abhängig. Kern des LIDAR-Systems bildet das TDOF-Messsystem, welches über eine Laser-Diode emittierte elektromagnetische Strahlung mittels eines Spiegelsystems ablenkt und über ein Sensorelement die reflektierte Strahlung detektiert (siehe Abbildung 2.11) [11, S.284].

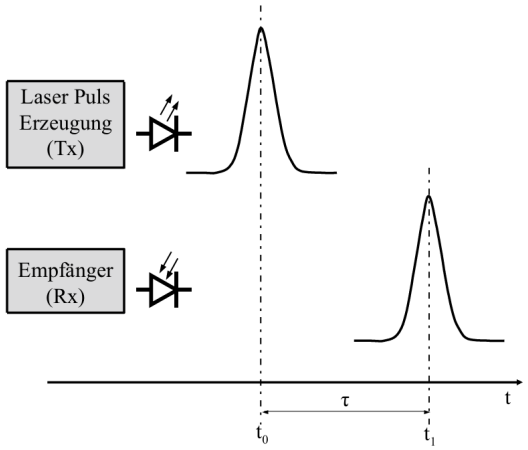
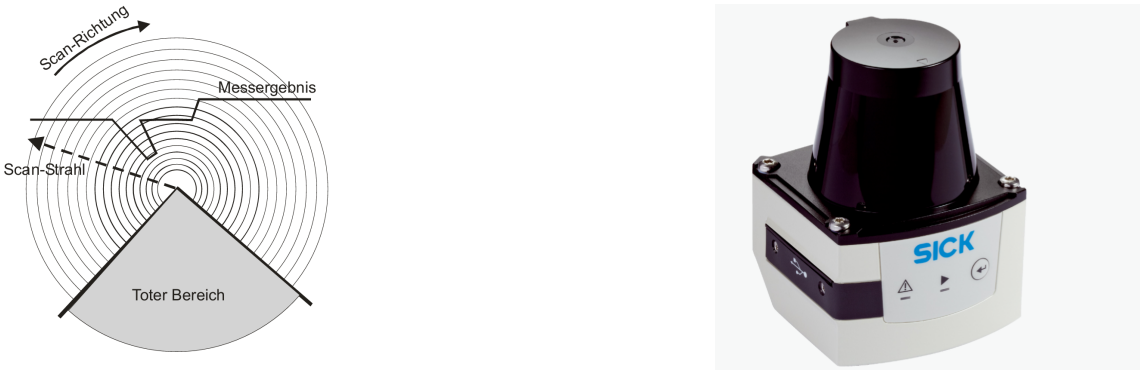


Abbildung 2.11: TDOF-Verfahren im LIDAR-System [11, S.291]

Auf Grund der Art und Weise der Datengenerierung, (siehe Abbildung 2.12a), ist es sinnvoll die erzeugten Daten in Polarkoordinaten zu verarbeiten, um diese verwenden zu können [11, S. 284]. Der in Abbildung 2.12a gezeigte „Tote Bereich“ ist abhängig vom LIDAR-System und nicht bei allen LIDAR-Systemen zwangsläufig vorhanden. Der Umfang des „Toten Bereichs“ ist hierbei primär von der Bauart abhängig.

Die aktive Distanzmessung wird mittels Laserstrahlung im Wellenlängenbereich zwischen 850 nm und 1 µm realisiert [50, S.320]. Um ein möglichst hohes Identifikationsvermögen der gesendeten Impulse zu gewährleisten, sollte der Messimpuls so kurz wie möglich gehalten werden [50, S.320].

Neben den Hardwarekomponenten eines LIDAR-Systems werden der Großteil der Signalauswertungsprozesse softwaretechnisch gelöst. Hierzu zählen unter anderem Abstands- und Relativgeschwindigkeitsauswertung. Diese weisen, je nach Hersteller, starken Varianzen im Funktionsumfang und in ihren Lösungsansätzen auf [50, S.322].



(a) Beispiel einer LIDAR-Messung

(b) Industrieller LIDAR-Sensor

Abbildung 2.12: Beispielabbildung für LIDAR-Systeme und -messungen [11, S. 284]

Der Empfangszweig des LIDAR-Systems ist ein maßgeblicher Performancefaktor, welcher in direkter Wechselwirkung mit den Umgebungseinflüssen des Sensors steht [50, S.321]. Je nach Anwendungsgebiet muss bei der Detektion der ausgesendeten Laserpulse eine Filterung der Eingangsgrößen realisiert werden, da hier durch äußere Lichtquellen, wie zum Beispiel die Sonneneinstrahlung, eine Störung der Messgrößen vorliegen kann. Diese Filterung wird hauptsächlich hardwareseitig durchgeführt [50, S.321]. Dies geschieht beispielsweise durch die Auswahl von geeigneten Receiverdioden. Hierzu zählen positiv intrinsic negative diode (PIN-Dioden) oder avalanche photodiode (APD-Dioden) [13]. Das eintreffende Signal wird nach einer Verstärkung dem sogenannten „Parallel-Gating“-Verfahren unterzogen. Im „Parallel-Gating“-Verfahren wird über einen Multiplexer eine Digitalisierung des Signals realisiert und in Range Gates abgelegt, welche interne Speicherorte darstellen [50, S.322]. Durch ein auftragen der eingehenden Sendeimpulse über die Zeit ergibt sich auf Grund der verschiedenen Intensitäten der eintreffenden Signale eine gaußsche Verteilung der Gates um den Messpunkt (siehe Abbildung 2.13). Durch das zusätzliche Anwenden stochastischer Mittel kann die Messgenauigkeit zusätzlich erhöht werden [50, S.322].

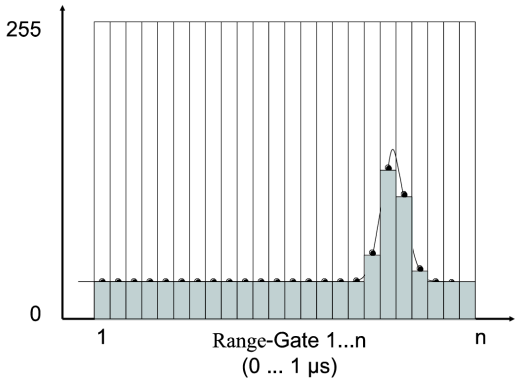


Abbildung 2.13: Schaubild parallel-gating [50, S.322]

Relevante Parameter für die erzielbare Leistung von LIDAR-Systemen stellen die Transmissions- und Refelxionseigenschaften des Ziels dar. Bei „On-Ground“ Anwendungen spielen neben Transmissons- und Reflexionseigenschaften auch die Dämpfungseigenschaften der Atmosphäre eine zentrale Rolle [50, S.323-324]. Diese Phänomene lassen sich in die drei Strahlungsphänomene Reflexion, Absoption und Transmisson einteilen. Die Koeffizienten dieser Phänomene sind abhängig von der Wellenlänge der Strahlung sowie der Temperatur und dem Material der bestrahlten Oberfläche. Ebenso sind diese Größen abhängig von den Zustandsgrößen des Mediums durch welches die Strahlung passiert [50, S.323-324]. Die Summe dieser Koeffizienten ergibt sich zu

$$\sigma_0 = \sigma_r + \sigma_{ab} + \sigma_t = 1. \tag{2.39}$$

Die Koeffizienten des Relexions-, Absorptions- und Transmissionsgrads lassen sich bestimmen zu,

$$\sigma_r = \frac{I_r}{I_0} \tag{2.40}$$

und

$$\sigma_{ab} = \frac{I_{ab}}{I_0} \tag{2.41}$$

und

$$\sigma_t = \frac{I_t}{I_0}. \tag{2.42}$$

Hierbei stellt σ_r die reflektierte σ_{ab} die absorbierte und σ_t die transmittierte Lichtintensität im Verhältnis zur ausgesendeten Intensität I_0 dar.

Die Auswahl der genutzten Wellenlänge ist somit direkt vom Einsatzbereich des LIDAR-Systems und der Hardwarekonfiguration abhängig und sollte den Umgebungsbedingungen angepasst werden.

2.2.2 Radio Detection and Ranging

Radio Detection And Ranging (RADAR) ist, wie das LIDAR, eine integrierte Sensorlösung um, je nach Messmethode, Geschwindigkeiten, Objektgrößen und Entfernungen zu bestimmen [11, S.280]. In integrierten Sensorsystemen wird zwischen dem Impuls-RADAR und dem Continous-

Wave-RADAR (CW-RADAR) unterschieden [11, S.280]. Impuls-RADAR-Systeme senden elektromagnetische Impulse in fest definierten Abständen aus, welche von einem im Sendeweg liegenden Objekt reflektiert werden. Über geeignete Verfahren, wie zum Beispiel dem TDOF, oder PDOA, kann aus diesen Messungen die Distanz und aus der Amplitude des Signals die Größe des reflektierenden Objekts berechnet werden. Die mathematischen Zusammenhänge für eine Laufzeitmessung mit Größenberechnung ergeben sich durch [11, S.281]

$$R = \frac{c}{2}\,\tau. \tag{2.43}$$

Hierbei stellt R die Größe des Objekts, c die Lichtgeschwindigkeit und τ die Laufzeit dar. Die Auflösung der Messung ist hierbei abhängig von der Impulsbreite [11, S.281].

CW-RADAR-Systeme bestehen aus einem gekoppelten Transmitter und Receiverpaar, welche konstant frequenzmodulierte Signale aussenden, um eine TDOF-Messung zu realisieren. Wenn sich Transmitter oder Receiver zusätzlich bewegen, kann über den Dopplereffekt ebenfalls die Geschwindigkeit bestimmt werden [11, S.281] siehe Kapitel 2.1.2. Der Aufbau eines Transmitter-Receiver-RADAR-Systems ist vereinfacht in Abbildung 2.14 dargestellt.

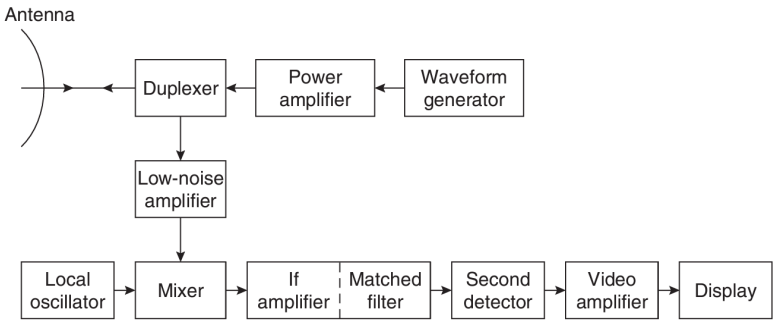


Abbildung 2.14: Schaubild Transmitter eines RADAR-Systems [45, S.20]

In Abbildung 2.14 wird der Transmitter über den in der Abbildung gezeigten Wellenformgenerator und Leistungsverstärker beschrieben, welcher frequenzmodulierte Signale erzeugt und diese im Anschluss verstärkt. Der Duplexer stellt ein Netzwerktor dar, welches die Antenne für Eingangssignale mit dem Receiver und für ausgehende Signale mit dem Transmitter verbindet (siehe Abbildung 2.15).

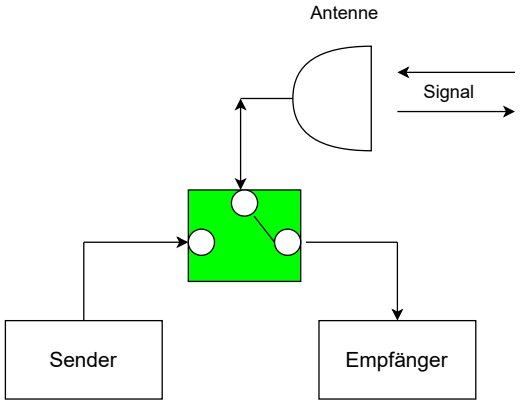


Abbildung 2.15: Schaubild der Funktionsweise eines Duplexers

Receiver von RADAR-Systemen sollten ein geringes Grundrauschen aufweisen, da dieses direkt mit der Leistungsfähigkeit und den Limitierungen des RADAR-Systems korreliert [45, S.3]. Die meisten RADAR-Systeme arbeiten im Mikrowellen-Frequenzbereich. In diesem Bereich des EM-Spektrums verursacht der erste Verstärker der Receiverschaltung den größten Rauschanteil, weswegen die Anforderungen an diesen, eine rauscharme Verstärkung zu gewährleisten, besonders hoch sind [45, S.4]. Ein weiterer maßgeblicher Faktor für die Leistungsfähigkeit des RADAR-Systems ist der Dynamikumfang des Receivers (siehe Abbildung 2.16) [45, S.4]. Der Dynamikumfang eines Verstärkers bezeichnet hierbei den Bereich (dieser wird in dB angegeben) in dem ein in ein Verstärker eingehendes Signal verstärkt werden kann [45, S.3]. Auf Grund des Phänomens des „Clutter-Effekts“ welcher dazu führt, dass durch Objekte in der Umgebung des RADAR-Receivers Echos erzeugt werden (was das Grundrauschen im empfangenen Signal erhöhen kann) ist der Dynamikumfang des Receivers von großer Bedeutung. Dieser ist am unteren Bereich durch das Grundrauschen limitiert und am oberen Bereich durch die Begrenzung beim Auftreten von Übersteuerung [45, S.3]. Begrenzung beim Übersteuern von verstärkten Signalen bezeichnet den Effekt an dem ein Signal nicht weiter verstärkt werden kann und somit die Amplitude nicht mehr im vollen Umfang abgebildet wird. [45, S.3]. Hierbei nimmt der Teil der Amplitude, welcher nicht mehr Abgebildet werden kann, das Erscheinungsbild einer geraden Begrenzung an.

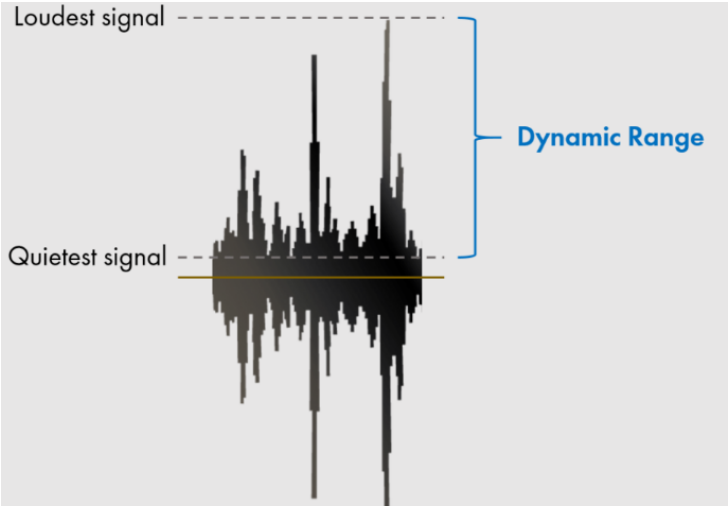


Abbildung 2.16: Schaubild Dynamikumfang [2]

Die Signalverarbeitung findet sich häufig im „IF“-Teil des Receivers und trägt die Aufgabe ungewolltes Rauschen aus dem empfangenen Signal zu filtern. Der Zweig der Schaltung welcher die Signalverarbeitung beinhaltet enthält ebenfalls den „matched-filter“, welcher das Signal-Rausch-Verhältnis maximiert und für die Detektionsentscheidung zuständig ist [45, S.3]. Ein Objekt gilt als detektiert, wenn der Ausgang des „IF“-Verstärkers einen vorher festgesetzten Grenzwert überschreitet. Ist dieser Grenzwert zu niedrig angesetzt, kann es fälschlicherweise zu Detektionen kommen. Die genaue Parametrisierung dieser Grenzwerte ist somit ein maßgeblicher Faktor in der Zuverlässigkeit des RADAR-Systems [45, S.3]. Des Weiteren kann das empfangene Signal über eine Anzeige zur Visualisierung ausgegeben werden, dieser Teil wird durch den „Video-Amplifier“ und das „Display“ dargestellt.

2.2.3 Stereo-Kameras

Stereo-Kamera-Systeme sind eine weit verbreitete Sensortechnik um mittels Triangulation (siehe Kapitel 2.1.5) Tiefeninformationen aus zwei zeitgleich aufgenommen Bildern zu extraprolieren. Der Aufbau einer Stereokamera ist im folgenden beschrieben (siehe Abbildung 2.17).

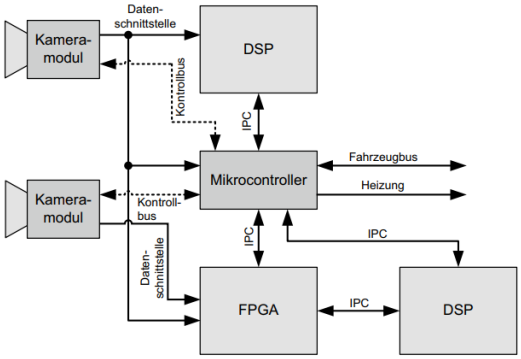


Abbildung 2.17: Schaubild Stereo-Kamera-System [50, S.364]

.

Für die Aufnahmefunktion von Bilddaten in Digitalkamera-Systemen stehen als Bildsensoren zwei Ansätze zur Auswahl: Das Charge Coupled Device (CCD) und der Complementary Metal-Oxide Semiconducter (CMOS), wobei sich in der Fahrzeugtechnik CMOS-Sensoren durchgesetzt haben [50, S.358]. Der Hauptvorteil dieser Sensorsysteme besteht darin, dass CMOS-Sensoren über aktive Pixel [50, S.358], sowie integrierte Analog-Digital-Wandler verfügen. Dies ermöglicht eine hochdynamische Aufnahmengenerierung, sowie die sensorinterne und -externe Weiterverarbeitung der generierten Bildsignale [50, S.358]. Die Aufnahmefunktion der Stereo-Kamera wird durch einen Microcontroller angesteuert, wobei beide Kameramodule eine Datenschnittstelle zu einem DSP-System und einem FPGA besitzen. Das DSP-System ist in der Regel optional und übernimmt, falls diese Funktionen vorgesehen sind, Aufgaben zur Objekterkennung [50, S.364]. Der im Schaubild gezeigte FPGA übernimmt die in Kapitel 2.1.5 gezeigten Berechnungen, um Tiefeninformationen aus den aufgenommenen Bildern zu gewinnen. Die Kommunikation zum übergeordneten System wird über den Mircrocontroller realisiert, wobei FPGA und DSP ebenfalls als schnelle Zwischenspeicher zur Datenverarbeitung zur Verfügung stehen [50, S.365].

Maßgeblich für die Genauigkeit von Stereo-Kamera-Systemen ist eine genaue Tiefenschätzung [50, S.365]. Relevante Faktoren für eine akkurate Tiefenschätzung bei achsenparallelen Stereo-Kamera-Systemen, sind wie im Kapitel 2.1.5 erwähnt, die genaue Erfassung der Brennweite f sowie des Abstandes b zwischen den Kameras [50, S.365].

Eine Erhöhung der Genauigkeit kann durch eine höhere Auflösung des Bildsensors, kleinere Pixelgrößen, einen größeren Abstand der Kameras zu einander oder eine größere Brennweite erzielt werden [50, S.365]. Veränderung dieser Parameter verändern jedoch wiederum relevante Systemparameter. Durch eine Vergrößerung der Brennweite wird beispielsweise ebenfalls das FOV der Kamera kleiner. Kleinere Pixelgrößen können zu einer geringeren Empfindlichkeit des Systems führen und den Bedarf der Rechenleistung zur Auswertung der Bildinformationen erhöhen [50, S.365]. Kalibrierungsverfahren können zusätzlich genutzt werden, um die Güte der erfassten Daten weiter zu verbessern [50, S.366]. Größen, die für die Kalibrierung eine maßgebliche Relevanz

haben, werden in intrinsische und extrinsische Kalibrierungsparameter unterschieden [50, S.366]. Zu den intrinsischen Parametern zählen, der Kamerahauptpunkt, Brennweite und Pixelskalierung. Zu den extrinsischen Parametern zählen die Kameraposition sowie die Kameraorientierung [50, S.366].

2.2.4 Globales Navigationssatellitensystem

Global Navigation Satellite System (GNSS) ist ein Sammelbegriff für Navigations-, beziehungsweise Positionsbestimmungsverfahren, mittels Satellitenkonstellationen. Die hierfür verwendete physikalische Methode zur Positionsbestimmung ist die in Kapitel 2.1.4 beschriebene Trilateration. Zu den am weitesten verbreiteten GNSS-Satellitensystemen gehören das GPS, Gallileo und GLONASS System [38].

GNSS-Systeme bestehen aus drei Hauptsystemen, dem „Weltraumsegment“, dem „Kontrollsegment“ und dem „Benutzersegment“ [38, S.4].

Das „Weltraumsegment“ bildet sich aus den GNSS-Satelliten welche mittels RF-Singalen untereinander und mit dem „Kontrollsegment“ kommunizieren [38, S.4]. Das „Kontrollsegment“ besteht aus Kommandozentralen, welche die Verfolgung der Position der Satelliten realisieren. Diese Kontrollzentren müssen immer eine Sichtlinie zu den zu verfolgenden Satelliten haben [38, S.4].

Das „Benutzersegment“ besteht aus einem oder mehreren Receivern, welche die gesendeten Signale der GNSS-Satelliten in Positions-, Geschwindigkeits- und Zeit-Approximationen des Receivers umrechnen [38, S.4].

Im Falle des GPS-Systems sind die GPS-Satelliten in 6 Orbitalebenen verteilt, um für jede Position auf der Erde zu gewährleisten, dass eine Sichtlinie zu mindestens 4 GPS-Satelliten besteht [38, S.4]. Die Grundlage hierfür bildet die eindeutige Bestimmbarkeit der Trilaterationmatrix, siehe hierfür Kapitel 2.1.4 [38, S.4].

Die RF-Frequenz von GPS-Satelliten sowie die der restlichen genannten Systeme operieren auf einem breiten Frequenzspektrum, welches Abbildung 2.18 zu entnehmen ist [38]. Zudem werden die gesendeten RF-Signale mittels Atomuhren zeitkodiert [38, S.6]. Die Informationskodierung erfolgt im Falle des GPS-Systems nach dem Binärsystem, wobei hierfür das Signal mit einer 180° Phasenverschiebung moduliert wird (siehe Abbildung 2.19) [38, S. 6].

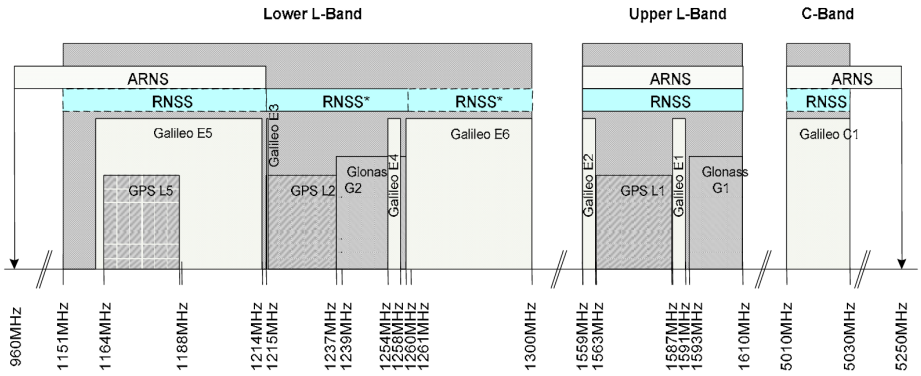


Abbildung 2.18: Frequenzbänder verschiedener GNSS-Systeme [38, S. 14]

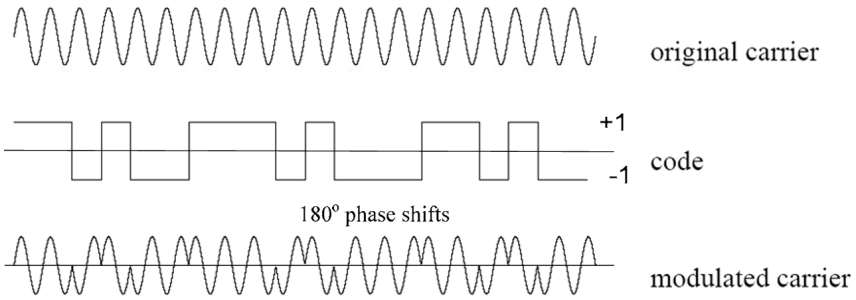


Abbildung 2.19: Frequenzmodulation eines RF-Signals mit 180 Grad Phasenverschiebung [38, S. 6]

Die Signalverarbeitung des übermittelten Signals erfolgt erfolgt im Receiver und dient der Extraktion der Navigationsinformationen aus dem Signal [38, S.14]. Auf Grund der Bewegung des Satellits während des Sendevorganges muss der Doppler-Effekt in der Signalverarbeitung berücksichtigt werden [38, S.14]. Um die Bewegung eines Satelliten zu beschreiben, ist es notwendig, zwei Referenzkoordinatensysteme zu definieren. Hierbei wird ein inertiales satellitenfestes Koordinatensystem definiert sowie ein terrestrisches erdfestes Koordinatensystem [38, S.16]. In der Satellitengeodäsie werden diese Koordinatensysteme genutzt, um die ermittelten Positionsparmeter vom satellitenfesten in das erdfeste Koordinatensystem zu transformieren [38, S.16]. Das erdfeste Koordinatensystem ergibt sich anhand des Conventional International Origin (CIO) [38, S.16]. In diesem stellt die z-Achse die Rotationsachse der Erde dar, die x-Achse liegt im Schnittpunkt des Äquators und dem Greenwich Meridians, welche beide mit der y-Achse ein konventionelles rechtshändiges System bilden [38, S.16].

Ein für die Genauigkeit von GNSS-Systemen entscheidendes Thema bilden Messfehler. Diese treten auf Grund verschiedener satelliteninterner Prozesse, sowie externer Prozesse auf und müssen in den Positionsberechnungen mit berücksichtigt werden. Im folgenden werden die verschiedenen möglichen Messfehler aufgelistet:

Satellite Clock Error

Dieser Fehler wird durch die Abweichung der internen Zeitgebung aufgrund fehlender Synchronisation des Satelliten zur wahren GPS-Zeit eingeführt [4, S.2] und lässt sich mittels der quadratischer Funktion

$$\delta^2 = a_B + a_D(t - t_0) + a_P(t - t_0)^2, \tag{2.44}$$

beschreiben, in welcher die Koeffizienten a_B, a_D und a_P, den Bias-, Drift- und Alterungs-Parameter der internen Atomuhr darstellen [4, S.2].

Orbital Error

Dieser Fehler ergibt sich durch Abweichungen in der angenommenen Orbitalbahn des Satelliten. Die „Orbital Error“ spielen in der heutigen Zeit eine vernachlässigbare Rolle, da die für die Positionsberechnungen notwendigen Satellitenbahnen nahezu in Echtzeit, mittels externer Dienste wie dem IGS (International GNSS Service), bestimmt werden können [38, S.22]

Multipath Error

Diese Fehler werden durch multiple Reflexionen der Signale am Receiver oder am Satelliten erzeugt, ähnlich des „Clutter-Effekts“ (siehe Kapitel 2.2.2). Vorbeugende Maßnahmen hierfür stellen ein vorteilhaftes Antennendesign dar, sowie das platzieren von Receivern an Orten, welche eine geringe Dichte an reflektierenden Objekten wie Bäume, Autos, Häuser aufweisen [38, S.23].

Receiver Clock Error

Der Receiver Clock Error entsteht durch geringere Genauigkeiten in den zeitgebenden Systemkomponenten des Receiver [38, S.22]. Dies lässt sich am Beispiel von GNSS-fähigen Mobilfunkgeräten beschreiben. Während die meisten Mobiltelefone zur Zeitmessung Quarzuhren verwenden, werden in Satelliten Atomuhren verwendet, welche geringe Laufzeitabweichungen ausweisen [38, S.6]. Der durch diese Differenz eingebrachte „Receiver Clock Offset“ muss in den Trilaterationberechnungen mit berücksichtigt werden. Außerdem werden aufgrund des „Receiver Clock Offsets“ vier anstelle von drei Satelliten für eine vollständige Bestimmbarkeit der Trilaterationsmatrix (siehe Kapitel 2.1.4), benötigt. Grund hierfür ist, dass durch die Berücksichtigung des „Receiver Clock Offsets“ eine weitere Unbekannte in der Trilaterationsmatrix eingeführt wird [38, S.23]. Das neue Gleichungssystem ergibt sich zu

$$\begin{bmatrix} r_R^1(t) \\ r_R^2(t) \\ r_R^3(t) \\ r_R^4(t) \end{bmatrix} = \begin{bmatrix} \sqrt{(x^1(t) - x_R)^2 + (y^1(t) - y_R)^2 + (z^1(t) - z_R)^2} + c\Delta\delta \\ \sqrt{(x^2(t) - x_R)^2 + (y^2(t) - y_R)^2 + (z^2(t) - z_R)^2} + c\Delta\delta \\ \sqrt{(x^3(t) - x_R)^2 + (y^3(t) - y_R)^2 + (z^3(t) - z_R)^2} + c\Delta\delta \\ \sqrt{(x^4(t) - x_R)^2 + (y^4(t) - y_R)^2 + (z^4(t) - z_R)^2} + c\Delta\delta \end{bmatrix}, \tag{2.45}$$

hierbei stellt $r_R^i(t)$ die Distanz zwischen Receiver$_R$ und Satellit^i dar. Die Indizierung der Koordinaten erfolgt analog zu diesem System. Hieraus ergeben sich bei bekannten Satellitenpositionen die Unbekannten x_R, y_R, z_R sowie $\Delta\delta$ [38, S.23], welche mittels vier unabhängiger Gleichungen eindeutig bestimmt werden können.

2.2.5 Startracker

Der Begriff „Startracker“ beschreibt ein Sensorsystem, welches anhand von astronomischen Konstellationen die Ausrichtung des Systems bestimmen kann in welchem der Sensor integriert ist [35, S.8]. Startracker sind die an häufigsten verwendeten und genausten Lagesensoren. Die Genauigkeit von Startrackern reicht von 1-3 arcsec bis zu 0,001 arcsec [35, S.8].

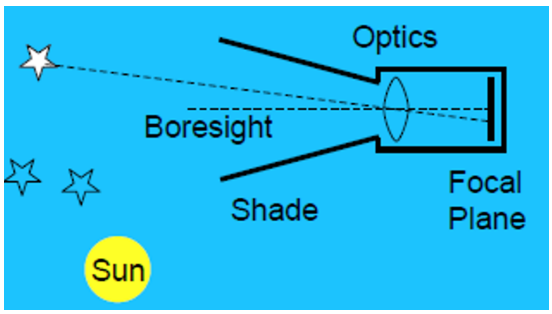


Abbildung 2.20: Aufbau eines Startrackers [35, S.8]

Ein Startracker besteht aus einer Bohröffnung, einem Objektiv und einer Fokusebene (siehe Abbildung 2.20). Die Fokusebene ist meistens als CCD umgesetzt, welche die aktive Detektion des einstrahlenden Sternenlichts realisiert. Das von einem Stern emittierte Licht wird von der CCD detektiert und über die relative Position des Bohrloches weiterverarbeitet. Hierbei werden Position und Intensität des einfallenden Lichts gespeichert und mit im Speicher des Systems abgelegten Sternkarten verglichen [35, S.8].

Zur Anwendbarkeit dieser Sensortechnik sind verschiedene Voraussetzungen zu erfüllen, zur genauen Identifikation der detektierten Konstellation bestehen zwei Ansätze (siehe Abbildung 2.21) [35, S.8].

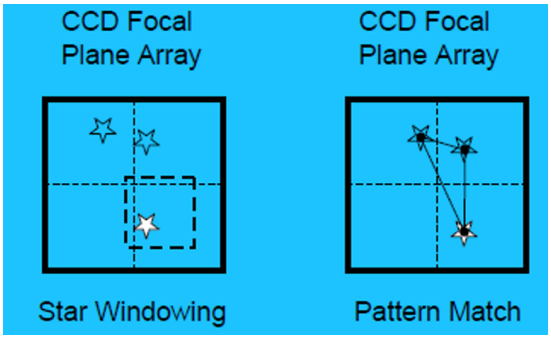


Abbildung 2.21: Pattern matching im Startracker [35, S.8]

Ansatz 1

Ist die Lage des Raumfahrzeuges innerhalb eines Ausrichtungswinkels von 0,5° bekannt, kann der Startracker zur Erhöhung der Genauigkeit der aktuellen Position genutzt werden. Dies ist möglich, indem durch die Randbedingung der ungefähren initialen Position eine Zuordnung der vom Startracker beobachteten Konstellation einfacher stattfinden kann, auch wenn dies alleine anhand der beobachteten Konstellation ohne das Wissen über die initiale Position nicht möglich wäre [35, S.8].

Ansatz 2

Die Lage des Raumfahrzeuges ist nicht genau bekannt, hier kann mittels eines Startrackers mit größerem FOV's aus dem detektierten Signal die genaue Lage des Raumfahrzeuges ermittelt werden [35, S.8].

Zur Sicherstellung der Funktionalität und Genauigkeit des Startrackers sind Störeinstrahlungen, wie durch Sonnenlicht, zu verhindern. Startracker kommen aufgrund des häufig beschränkten FOV's hauptsächlich in dreiachsig stabilisierten Raumfahrzeugen zum Einsatz [35, S.8].

Abbildung 2.22: ASTRO APS Startracker von Jena Optronik [35, S.8]

Startracker sind in implementierten Sensorsystemen in drei Auslegungen vertreten [47, S.2].

Star Scanner

Star Scanner stellen Startracker dar, welche an der Außenseite eines rotierenden Raumfahrzeuges montiert sind. durch die Drehbewegung wird ein „Scan“ der umliegenden Sternenkonstellationen realisiert [47]. Das Objektiv ist mit einer Schlitzblende versehen, um einen fest definierten Bereich des Aufnahmesensors in einem gewissen Moment zu exponieren. Hauptfaktoren, welche die Genauigkeiten der Aufnahmen beeinflussen, sind die Anzahl der beteiligten Rotationsachsen, sowie die Gleichförmigkeit der Rotationsbewegung und die Größe des Schlitzes [47].

Gimballed Star Trackers:

Diese Art des Startrackers wird vor allem bei Raumfahrzeugen eingesetzt, welche eine große Varianz in ihren Orbithöhen und Ausrichtungen haben [47, S.3]. Der Startracker ist in dieser Ausführung auf einem Gimbal montiert, um seine Ausrichtung und Position am Raumfahrzeug dynamisch zu verändern und das effektive FOV zu erweitern und somit komplexere Navigationsmanöver zu realisieren [47, S.3]. Auf Grund der erhöhten Komplexität des Systems und der größeren Anzahl an beweglichen Komponenten ist die Lebensdauer des „Gimballed Star Trackers“ in der Regel kürzer als bei konventionellen Startrackern [47, S.3]. Diese Form von Startrackern kommt vor allem bei stationären, nicht rotierenden Satelliten zum Einsatz, wie zum Beispiel geostationäre Satelliten [47, S.3].

Head Star Trackers:

Diese Form von Startracker ist eine Mischform der vorherig genannten. Die Such- und Verfolgungsfunktion des Startrackers ist in diesem intern in den Sensor integriert. Hierbei generiert das einfallende Licht der beobachteten Konstellation ein Ladungsmuster auf das bildgebenden Sensorelement (siehe Abbildung 2.23).

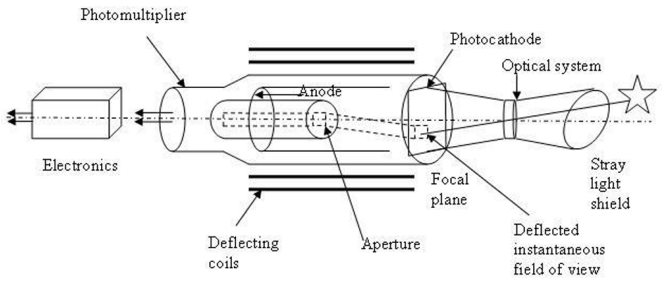


Abbildung 2.23: Abbildung eines Head Star Trackers [47, S.4]

Dieses Muster wird eingelesen und und mittels DSP verarbeitet. Anstatt den Startracker oder das Raumfahrzeug zu bewegen, werden die generierten Ladungsmuster verarbeitet und mit Kon-

stellationen einer internen Datenbank verglichen, bis das generierte Ladungsmuster nicht mehr detektiert werden kann. Wenn dieser Fall eintrifft startet der Prozess erneut [47].

2.2.6 Sonnensensoren

Sonnensensoren sind eine weitere in der Raumfahrt häufig verwendete Technologie, um eine Ausrichtungsbestimmung zu realisieren. Sonnensensoren benutzen exponierte Fotosensoren, um die Richtung der einfallenden Sonnenstrahlung zu messen [35]. Hierbei werden häufig Pyramidenkonfigurationen verewndet, wie in Abbildung 2.24 dargestellt.

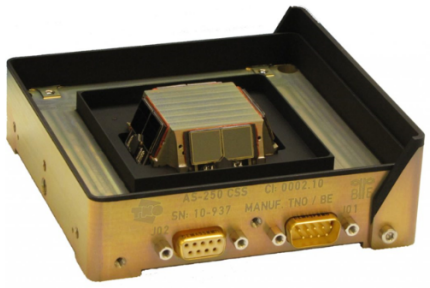


Abbildung 2.24: Abbildung einer Pyramidenkonfiguration von Fotosensoren [6]

Die Intensität der einfallenden Sonnenstrahlen ist hierbei abhängig vom Kosinus des Einfallwinkels normal zur Oberfläche des Fotosensors [35, S.3]. Der Hauptvorteil der Pyramidenkonfiguration besteht in der Realisierung einer höheren Genauigkeit. Hiermit können Genauigkeiten von 0,1° bis 0,01° erreicht werden [35, S.3]. Die Messung mittels Sonnensensor ermöglicht allerdings nur eine Richtungsbestimmung, somit ist eine weitere Messung notwendig, um eine genaue Lagebestimmung zu realisieren. Dies kann mittels einer zweiten Messung des gleichen Sensors mit relevantem Zeitabstand zur ersten Messung geschehen, oder mittels einer weiteren Sensortechnologie, wie zum Beispiel eines IMU-Systems (siehe Kapitel 2.2.8) [35, S.3].

Um die Präzision des Sensorsystems weiter zu erhöhen, kommen in Sonnensensor-Systemen häufig Schlitzkammerkonfigurationen zum Einsatz.

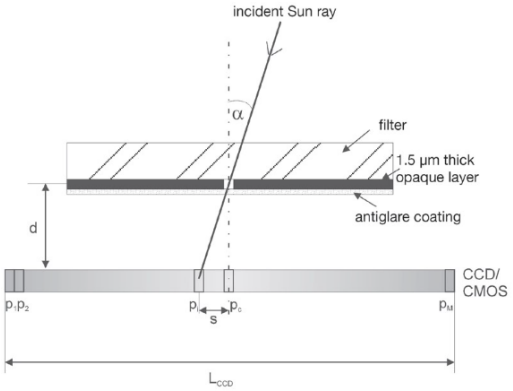


Abbildung 2.25: Abbildung Schlitzkammerkonfiguration im Halbschnitt [16]

In Schlitzkammerkonfigurationen (siehe Abbildung 2.25) reduziert sich das einfallende Sonnenlicht durch die Schlitzblende zu einer dünnen Linie, welche auf den Boden der Schlitzkammer projiziert wird. Der Boden der Schlitzkammer ist mit einem Netzwerk aus Fotozellen versehen, welche verschiedene Größen und Ausrichtungen aufweisen [35, S.4]. Werden zwei dieser Schlitzkammersensoren orthogonal zueinander positioniert, kann eine eindeutige Bestimmung des Richtungsvektors des Raumfahrzeuges realisiert werden [35, S.4].

2.2.7 Magnetometer

Magnetometer kommen zur Positionsbestimmung von erdnahe Satelliten zum Einsatz. Magnetometer werden genutzt, um durch eine Magnetfeldmessung die Ausrichtung des Satelliten im Verhältnis zur Erde zu bestimmen. Das Funktionsprizip des Sensors basieren auf der Feldstärken- und Richtungsmessung, welche aus den Änderungsraten der Feldstärken und Richtungsgrößen ermittelt werden. Da Magnetometer in einer großen Anzahl an verschiedenen Auslegungen vorkommen, werden im Folgenden einige der verbreitetsten erläutert [11, S.440].

Hallsensoren

Hallsensoren bilden eine der weitverbreitetsten implementierten Sensorlösungen zur Messung magnetischer Feldstärken ab [11, S.374]. Ein Hallsensor besteht aus einem Halbleiterplättchen, an welchem eine Spannung abgenommen werden kann. Wirkt auf diese eine magnetische Feldgröße, ist die gemessene Polarisierung der Spannung und deren Größe direkt proportional zur Feldstärke sowie der Polarisierung des Magnetfeldes [11, S.439].

GMR-Sensoren

Funktionsgrundlage der GMR-Sensoren ist der GMR-Effekt. Dieser tritt in dünnschichtigen Lagen aus ferromagnetischer und nichtmagnetischer Materialien auf. Der elektrische Widerstand, welcher über den Sensor zu messen ist, ist hierbei direkt abhängig von der Polarisationsrichtung der Schichten, welche durch äußere Magnetfelder beeinflusst werden können [11, S.439]. Diese Sensortechnologien finden vorrangig Anwendungen in Lese-Schreib-Köpfen von HDD-Festplatten. [11, S.439].

Feldplatten

Dieser Sensortyp nutzt das Prinzip der Ladungsverteilung innerhalb eines elektrischen Leiters auf welchen ein Magnetfeld wirkt, um magnetische Feldgrößen zu messen [11, S.439]. Hierbei ändert sich der gemessene Widerstand über die Feldplatte in Abhängigkeit der Feldstärke und Richtung des Magnetfeldes [11, S.439].

Als technisch nutzbare Implementierung von Magnetometern in Raumfahrzeugen zu Positions- und Richtungsbestimmung werden drei orthogonal zueinander gelegene Magnetometer verwendet (siehe Abbildung 2.26) [35, S.7].

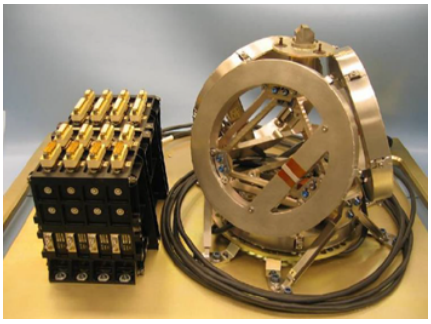
Abbildung 2.26: Beispielabbildung eines Magnetometers von ZARM [52]

Zur korrekten Messung und Interpretation der drei Komponenten des Magnetfeldes ist als Referenz das IGRF notwendig. Zur Zuordnung zu diesem wird eine Anfangsposition benötigt, welche über ein zweites Sensor-System generiert werden muss [35, S.7]. Durch die Bestimmung des Feldvektors, kann mittels geeigneter Koordinatentransformationen die Ausrichtung des Raumfahrzeugs zum externen Magnetfeld bestimmt werden [35, S.7]. Die Anwendbarkeit von Magnetometern zur Lage- und Positionsbestimmung ist auf planetennahe Umlaufbahnen beschränkt. Voraussetzung ist, dass der Himmelskörper ein Magnetfeld besitzt und die Beschaffenheit und Form des Magnetfeldes ausreichend genau modelliert werden kann [35, S.7]. Analog zum Sonnensensor (siehe Kapitel 2.2.6) generiert die Messung mittels Magnetometer nur einen Richtungsvektor, zur genauen Positions- und Richtungsbestimmung ist ein zweiter Sensor erforderlich,

um diese Größen ganzheitlich zu bestimmen [35, S.7]. Die Genauigkeit von Magnetometern im Kontext der Positionsbestimmung korreliert direkt mit der Imperfektion in den Modellen zum Erdmagnetfeld. Hierbei können Genauigkeiten von bis zu 0,5° am Äquator und 3° an den Polen realisiert werden [35, S.7].

2.2.8 Inertiale Messeinheiten

Inertiale Messeinheiten (IMU's) (siehe Abbildung 2.27a) verwenden verschiedene inertiale Sensoren um Geschwindigkeits-, Positions- und Orbithöhen-Messungen zu realisieren [35, S.10]. Sie stellen somit Sensorsysteme dar, welche aus verschiedenen abgegrenzten Sensorsystemen bestehen. Hierbei wird häufig folgende Kombination verwendet, drei Beschleunigungssensoren in orthogonaler, lineare und einfacher Auslegung, sowie drei Gyroskope zur Ausrichtungsbestimmung und Richtungsoptimierung.



(a) Beispieldarstellung einer IMU

(b) Litton LN-200 IMU

Abbildung 2.27: Abbildung zweier Kommerziell eingesetzten IMU-Systeme [35, S.10]

Die Abweichungen der Geschwindigkeits- und Ausrichtungsmessung befindet sich in Raumfahrtanwendungen für kommerzielle IMU's bei 1,6 km/h und 0,1 °/h [35, S.10]. IMU's können in verschiedenen Ausführungen hergestellt werden. Die Hauptunterschiede zwischen verschiedenen IMU-Systemen bestehen in der verwendeten Sensortechnologie zur Umsetzung von Beschleunigungs- und Drehraten-Messung. Diese reichen von klassischen Gyroskopen, zu Fasergyroskopen (siehe Abbildung 2.27b *Litton LN-200*) [31], IC basierte Gyroskope, klassischen seismischen Massenschwingern, DMS-Systemen und IC-basierten Beschleunigungssensoren [54]. Wie bereits in Kapitel 2.1.6 dargestellt, lassen sich mittels der detektierten Wegänderungen über die Zeit Aussagen über Ort, Geschwindigkeit und Position treffen. Die auftretende mittlere quadratische Abweichung steigt mit der Iterationszahl der Berechnungen der Position, was zu Abweichungen von der tatsächlichen Position führen kann [54]. Aus diesem Grund sind die Messungen nur in einem kurzen Zeitraum in ihrer Genauigkeit konsistent. Um die Genauigkeit der Messung über die Zeit zu erhöhen, ist es notwendig die sich summierende mittlere quadratische Abweichung durch zusätzliche Informationen zu korrigieren. Dies kann durch eine Lokalisierung

mittels GNSS (siehe Kapitel 2.2.4), oder mittels einer Ausrichtungsmessung durch Startrackersystemen (siehe Kapitel 2.2.5) ermöglicht werden.

Micro-Electromechanical Systems (MEMS) erlangen vor allem im Kontext von Cubesat-Systemen immer mehr an Bedeutung [54]. Grund dessen ist vor allem ihre geringe Baugröße, hohe Sensitivität und geringen Produktionskosten. Innerhalb dieser Sensorsysteme sind Beschleunigungssensoren und Gyroskope als IC-Systeme umgesetzt. Nachteil dieser Systeme war lange ihre niedrigere Genauigkeit im Vergleich zu beispielsweise faseroptischen Gyroskopen [54, S.1]. Diese Unterschiede wurden in den letzten Jahren durch technologische Weiterentwicklungen jedoch immer geringer [54], weswegen MEMS im heutigen Kontext eine immer bedeutsamere Rolle in Raumfahrtsystemen einnehmen. Das Messprinzip von MEMS-Sensorsystemen basiert auf einer kapazitiven Messung, wobei die seismische Masse und der Sensorkörper von einander isoliert sind. In dieser Konfiguration bilden der Sensorkörper und die seismische Masse die Kodensatorplatten über die die Ladung bzw. die Kapazität des MEMS-Systems bei Auslenkung gemessen werden kann (siehe Abbildung 2.28) [11, S.266].

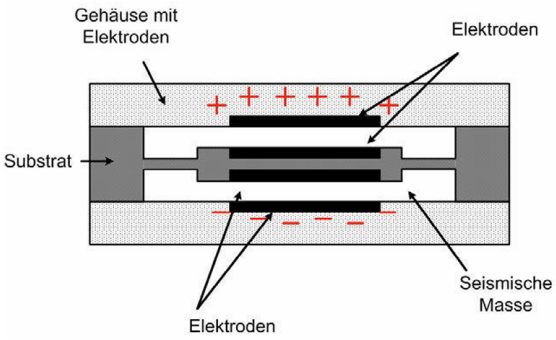
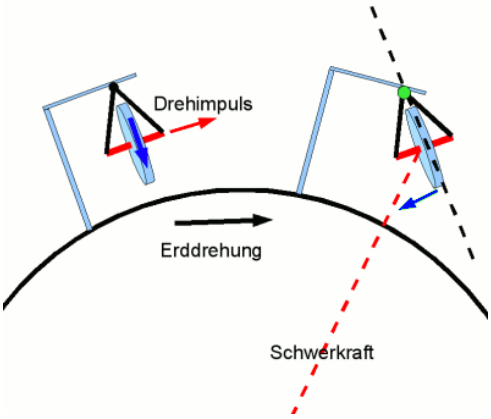
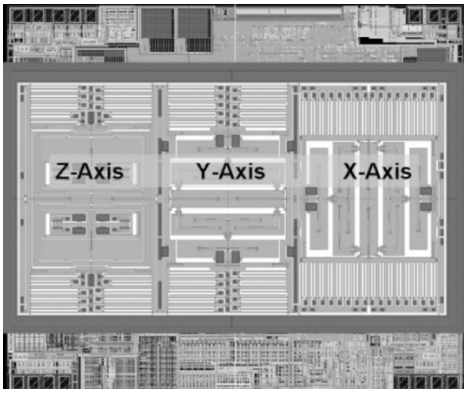


Abbildung 2.28: Abbildung eines MEMS-Systems [11, S.266]

Das Funktionsprinzip von gyroskopischen Messsystemen lässt sich wie folgt beschreiben. Ein Gyroskop besteht aus einer schwingenden oder rotierenden seismischen Masse. Sobald auf diese bewegte seismische Masse ein äußeres Drehmoment wirkt, entsteht eine Reaktionskraft. Diese Reaktionskraft wird durch die Coriolis-Kraft dargestellt, welche die seismische Masse auslenkt [11, S.374]. Die aus dieser Kraft resultierende Auslenkung wird in Gyroskopen als Messgröße erfasst. Das hierbei bekannteste Messverfahren wird durch den Kreiselkompass dargestellt, welcher in Abbildung 2.29a zu sehen ist [11, S.374]. Des Weiteren existieren ebenfalls für gyroskopische Messprinzipe MEMS-Sensoren, welche in Form schwingender seismischer Massen ausgelegt werden [11, S.375]. Eine Beispiel für ein MEMS-Gyroskop ist dabei Abb. 2.29b zu entnehmen.



(a) Beispieldarstellung eines Kreiselkompasses



(b) MEMS Gyroskop

Abbildung 2.29: Abbildung eines MEMS-Gyroskops innerhalb eines IC's [11, S.375]

2.2.9 Auto-Ident-Systeme

Auto-Ident-Systeme, oder auch automatische Identifikationssysteme werden genutzt, um anhand verschiedener Charakteristiken Objekte zu identifizieren. Die am häufigsten verwendeten Technologien stellen Barcode-laserscanner, Auto-Ident-Kameras und RFID-Lesegeräte dar [11, S.285]. Auch wenn es immer wieder Innovationen und Neuerungen in diesem Bereich der Sensortechnologie gibt, stellen diese Sensoren die meist verwendeten Auto-Ident-Systeme dar [11, S.285]. Im Folgenden werden die oben genannten Sensor-Systeme im Detail erläutert.

Bar-/QR-Code Scanner

Bar-/QR-Code Scanner verfügen über ein optoelektronisches Bauteil, welche die empfangenen Bilddaten erzeugt [11, S.286]. Die hauptsächlich zum Einsatz kommenden Sensoren sind wie in Kapitel 2.2.3 bereits erwähnt, CCD- oder CMOS-Sensoren sowie Scanner mit Laserabtastung, auch „flying spot scanner“ genannt [11, S.286]. Die Laserimpulse werden über Halbleiterdioden erzeugt [11, S.286]. Die Wellenlänge der Laserimpulse befindet sich im roten Spektrum, ähnlich wie beim LIDAR (siehe Kapitel 2.2.1) [11, S.286]. Hierbei werden Sendeleistungen von 1 mW bis 10 mW verwendet. Die Bauprinzipien in Bar-/QR-Code Scanner unterscheiden sich in Rundum-Scanner und Scanner nach dem Autokollimationsprinzip [11, S.286]. Bar- und QR-Code Scanner nutzen hierbei Binärisierer, um die eingelesenen Bildsignale in Binärcode umzuwandeln. Diese können darauf hin von einem Decoder Interpretiert und über die anliegende Schnittstelle ausgegeben werden [11, S.287].

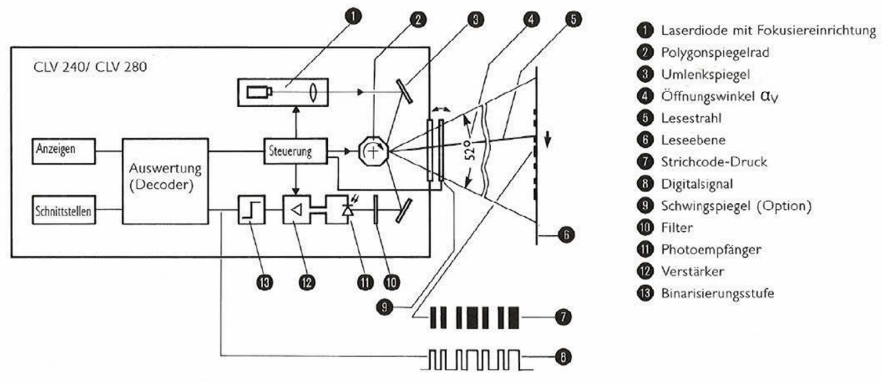


Abbildung 2.30: Aufbau eines Bar-Code Scanners (Erzeugung des Laserstrahls und Aufbau der Optik sowie FOV können Abweichen) [11, S.287]

Die Genauigkeit der Messung des Bar-/QR-Code Scanners hängt unter anderem von der Entfernung zum Ziel, ob ein implementierter Autofokus verbaut ist, sowie vom Winkel der Aufnahme ab und ist für jedes Gerät durch Herstellerangaben spezifiziert [11, S.291-292].

RFID-Lesegeräte

RFID ist ein Akronym für Radiofrequenzidentifikation. RFID-Systeme (siehe Abbildung 2.31) bestehen aus einem Datenträger, welcher sich am zu identifizierenden Gegenstand befindet und einen spezifischen Code enthält, sowie einem Lesegerät, welches in der Lage ist den im Datenträger befindlichen Code zu interpretieren [11, S.296]. Der Datenträger, auch Tag genannt, beinhaltet die vom Lesegerät auszulesenden Informationen und wird von diesem beim Leseprozess aktiviert. Zudem können die im Tag gespeicherten Informationen beim Auslesen aktualisiert werden [11, S.296]. RFID-Tags werden in der Regel in zwei Gruppen unterschieden, in aktive und passive RFID-Tags [11, S.296]. Passive RFID-Tags haben keine eigenen Stromversorgung und werden induktiv durch die RF-Signale des Lesegeräts versorgt. Aktive RFID-Tags verfügen über eine eigene Stromversorgung. Diese Tags verfügen über höhere Reichweiten sind jedoch in der Produktion wesentlich kostenintensiver [11, S.296].

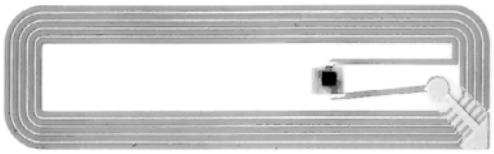


Abbildung 2.31: Abbildung einer RFID-Antenne im Frequenzbereich 13,56 MHz [11, S. 297]

RFID-Systeme können in verschiedenen Frequenzbändern verwendet werden, hierzu zählen **LF-**, **HF-**, **UHF-** und **SHF-**Bänder [11]. Hierbei gilt näherungsweise, umso größer die Frequenz des RFID-Tags ist, desto größer ist die wirksame Distanz in welcher dieser ausgelesen werden kann [11].

Auto-Ident-Kameras

Auto-Ident-Kameras nutzen ähnlich wie Bar-/QR-Code Scanner Bildsensoren, um Daten aus aufgenommenen Bildern auszulesen. Die im Bildsensor aufgenommenen Daten werden hierbei an einen FPGA geleitet und hier vorverarbeitet. Dadurch wird die in der Prozessierungskette nachfolgende CPU von Vorverarbeitungsarbeiten, wie Filteroperationen und Binärisierung, entlastet [11, S.293]. Die im CPU verarbeiteten Daten werden über eine Schnittstelle (Ethernet, Seriell oder CAN) ausgegeben und können dann weiter verarbeitet werden [11, S.293]. Auto-Ident-Kameras werden häufig zur Produkt/Objekt-Identifizierung verwendet und sind maßgeblich durch die Sichtbarkeit der zu detektierenden Markierungen begrenzt [11, S.293].

3 Das Testbett

Das folgende Kapitel dient der Beschreibung des Testbetts. Ziel bei der Entwicklung des Testbetts ist es einerseits die für die Lokalisierung im Nah- und Fernbereich, sowie die autonome Annäherung nötige Hardware zu implementieren. Zudem ist es Ziel ein Software Framework zu entwickeln, welches die Nutzung, sowie Verarbeitung der Daten ermöglicht. Das angestrebte Software Framework, soll außerdem die Funktionalität der Lokalisierung und autonomen Annäherung gewährleisten. Die sich daraus ableitenden Bereiche werden wie folgt in diesem Kapitel beschrieben: Erläuterung des Grundkonzepts des Testbetts, mechanischer Aufbau, Hardware des Testbetts, Benchmarking der Sensorik, Software des Testbetts und Integration der Hardware und Software.

3.1 Grundkonzept

Das Grundkonzept des Testbetts ist es eine Plattform zu schaffen, welche einerseits die zur Lokalisierung im Nah- und Fernbereich notwendige Sensorik und Software beinhaltet, manuell steuerbar ist und die Datenverarbeitung der Sensorinputs realisieren kann. Diese Voraussetzungen wurden vom DLR Bremen für die Masterarbeit gestellt. Das Testbett soll Grundlage sein um andere, von dieser Abschlussarbeit unabhängige, Hardware- und Software-Komponenten welche im Kontext von „Systems of Systems“-Anwendungen notwendig sind auf dieser zu Integrieren und später zu testen. Diese sollen im späteren Verlauf, allerdings ebenfalls unabhängig von dieser Abschlussarbeit, in Szenarios des Rendezvous und Dockings sowie dem Austauschen Daten und Systemattributen zusätzlich getestet werden. Hauptaufgabe in der Konzeptionierung des Testbetts war es Randbedingungen festzulegen, welche eine Abbildbarkeit des späteren Anwendungsfalles unter terrestrischen Bedingungen ermöglicht. Diese Randbedingungen wurden durch das DLR zur Orientierung während des Projektverlaufs gestellt.

Diese Randbedingungen leiten sich aus, den in Kapitel 1 genannten, Phasen IV und V der 5 Phasen des Dockings ab [7]. Zur Umsetzung der „Far-Rendezvous-Operation-Phase“ und „Close-Rendezvous-Operation-and-Spacecraft-Mating-Phase“ [7] ist es notwendig Informationen über die relativ und absolute Position der am Rendezvous und Docking beteiligten Systeme zu generieren. Aus dieser Anforderung ergeben sich die im Folgenden beschriebenen Randbedingungen, welche notwendig für die Abbildbarkeit des Rendezvous- und Docking-Problems im Testbett sind. Neben den Randbedingungen stellen sich Grundvoraussetzungen zum Entwicklungskonzept welche projektbedingt und vom DLR gestellt sind. Hierbei stellen die Randbedingungen

den Funktionsumfang und die Vereinfachungen dar, welche während der Entwicklung des Testbetts angenommen werden und leiten sich aus den Grundvoraussetzungen ab. Die Grundvoraussetzungen stellen die übergeordneten Attribute des Testbetts dar welche in diesem abgebildet werden sollen.

Randbedingung 1

Eine dieser Randbedingungen sind die Freiheitsgrade, welche das Testbett bezüglich seiner Bewegung aufweist. Raumfahrzeuge weisen sechs Freiheitsgrade auf, drei translatorische sowie drei rotatorische. Auf Grund der Tatsache, dass nicht die exakte Abbildung der Bewegung von Systemen mit 6 Freiheitsgraden und deren Regelung Ziel des Testbettes ist. Sondern die Sensorik und deren Datenverarbeitung im Vordergrund steht, gilt es die Freiheitsgrade des Testbetts auf zwei Freiheitsgrade zu reduzieren. Diese Randbedingung leitet sich direkt aus der Simplifizierung des Bewegungsspektrums von am Rendezvous und Docking beteiligten Systemen ab. Des weiteren ermöglicht diese Randbedingung eine Abbildbarkeit dieses Problems im zweidimensionalen Raum.

Randbedingung 2

Eine weitere Randbedingung des Testbetts ist es für die Lokalisierung und Annäherung notwendige Sensorik und Aktuatorik und die damit verbundene Datenverarbeitung innerhalb eines erweiterbaren Softwareframeworks zu verbinden. Ziel hierbei ist es die von Sensorik und Aktuatorik generierten und benötigten Daten einheitlich zu verarbeiten, weiter zu gegeben und zu analysieren.

Randbedingung 3

Weiter stellt sich die Randbedingung dem Testbett zu ermöglichen mittels Sensorik den Raum in welchem sich dieses bewegt zu erfassen und mittels dieser Daten Navigationsaufgaben zu lösen. Zusätzlich soll das Testbett manuell steuerbar sein und in der Lage sein innerhalb einer Büroumgebung manövrieren zu können.

Grundvoraussetzung 1

Grundvoraussetzung ist zudem das die Systemkomponenten in ihrem Design über ein hohes Maß an Flexibilität verfügen um Änderungen und Anpassungen innerhalb des Testbettes schnell ermöglichen können.

Grundvoraussetzung 2

Zweite Grundvorraussetzung des Testbetts ist es eine Abbildbarkeit der Phasen IV und V des Rendezvous und Dockings zu bieten, welche auf eine Büroumgebung skalierbar sein muss. Zur Skalierung und Definition der Abstandsbereiche für Nah- und Fern-Bereich wird die in Anhang A.1.4 aus dem ESA Projekt LIRIS gezeigte Definiton (siehe Abbildung A.6) verwendet. Wie in

[27, S.2] definiert ergeben sich diese Bereiche zu 40km - 3,5km zum Fernbereich und 3,5km - 0km im Nahbereich. Somit ergibt sich ein Skalierungsfaktor von

$$sk = \frac{3{,}5km}{40km} = 0{,}0875. \tag{3.1}$$

Die maximale gradlinige Distanz im Büroraum ergibt sich zu 7,5m, daraus ergibt sich eine skalierte Einteilung in den Fernbereich von

$$7{,}5m \cdot 0{,}0875 = 0{,}656m. \tag{3.2}$$

Der daraus resultierende skalierte Fern- und Nahbereich ergibt sich zu 7,5m bis 0,65m und 0,65m bis 0m.

Um Nahbereichsmanöver durchzuführen wird hierbei eine Positionsgenauigkeit von $\approx$ 0,01m in x- und y-Richtung vorausgesetzt.

3.2 Hardware des Testbettes

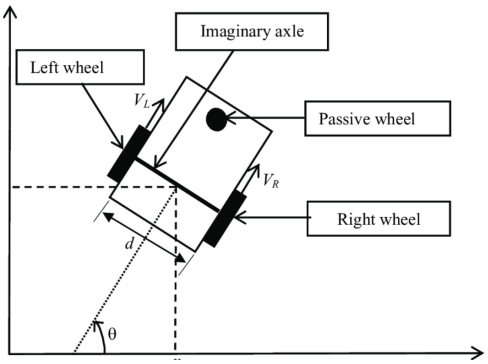
Im Angesicht der Hardwareauswahl ist die Betrachtung der für das Testbett bestimmten Randbedingungen sowie der in Kapitel 2.2 dargelegten Sensorik entscheidend, um auf Grundlage dieser ein Auswahl für die notwendigen Komponenten zu treffen. In den folgenden Unterkapiteln werden diese Aufgeführt und differenziert betrachtet um auf Grundlage dieser Aspekt eine Hardwareauswahl zu treffen. Ausgeschlossen von einer differenzierten Betrachtung sind jene Komponenten, welche als Vorgabe durch das DLR gestellt wurden. Grund für diese Vorgaben sind Beispielsweise, das Vorhandensein von in Vergangenheit beschaffter Materialien, oder Materialien und Sensoren, welche als projektkritisch durch das DLR eingestuft wurden.

3.2.1 Antrieb

Nach den in Kapitel 3.1 durch das DLR gestellten Randbedingungen können folgende Punkte bezüglich des Antriebsystems festgehalten werden. Das Antriebssystem muss in der Lage sein das Testbett eine freie Bewegung im zweidimensionalen Raum zu ermöglichen. Das Antriebssystem muss somit mindestens zwei Freiheitsgrade zur Verfügung stellen. Durch bereits in vorherigen Projekten des DLR's beschafften Mitteln standen für die Konzeptionierung des Antriebs bereits zwei Elektromotorsysteme zur Verfügung. Diese Elektromotoren, welche für ein differential Antriebssystem (siehe Abbildung 3.1b), von der Firma Parallax konzipiert wurden (siehe Abbildung 3.1a) wurden als Vorgabe durch das DLR festgehalten. Das vorgegebene Antriebssystem umfasst zusätzlich optische Encoder, sowie eine Übersetzung mittels Schneckengetriebe. Bei den Enco-

dern handelt es sich um inkrementale Quadratursignal Impulsgeber, mit einer Radauflösung von 36 Zählungen pro Umdrehung, welche eine Auflösung von 144 Impulsen pro Umdrehung bietet [36, S.12]. Somit kann mit den Encodern eine Auflösung von 2,5° pro Impuls realisiert werden.

(a) Beispieldarstellung der Parallax Motoren



(b) Abbildung der Freiheitsgrade im Differentialantriebssystem

Abbildung 3.1: Abbildungen des Differentialantriebs [36]

Bei den Verbauten Motoren handelt es sich um Gleichstrom Bürstenmotoren [36, S.1]. Die Nominalspannung zum betreiben der Motoren ohne Last bei 100 RPM liegt bei 12V und 1,5A. Die Steuerung der Motoren Erfolgt über Pulsweiten modulierte Signale, wobei über die Pulsweite die Rotationsgeschwindigkeit der Räder eingestellt werden kann [11, S.349]. Die aus Aluminium gefertigte Motoraufnahme des Antriebssystems kann eine Nutzlast von bis zu 27,2kg tragen [36]. Bei den verwendeten Rädern handelt es sich um pneumatische Räder, welche mit einem Schlauch ausgestattet sind und über Aluminiumfelgen mit der Achse verbunden werden [36]. Der Raddurchmesser der profilierten Reifen beträgt 6 Zoll (15,24cm) [36]. Eine Explosionszeichnung des Antriebssystems ist dem Anhang in A.2 zu entnehmen. Die im Antriebssystem verbauten optischen Encoder basieren auf optischen Quadratursignalmessungen, welche von einer Steuerung abgetastet, digitalisiert und interpoliert werden können um eine Winkelmessung zu realisieren [11, S.216]. Bei den verwendeten Encodern handelt es sich um zweikanälige Encoder, welche jeweils zwei um 180° phasenverschobene Quadratursignale erzeugen [36]. Durch Messungen am jeweiligen Start und Ende der einzelnen Quadratursignale kann die Auflösung durch die Phasenverschiebung um den Faktor 4 erhöht werden [11, S.217]. An der Achse befindlich ist in diesem Antriebssystem eine Kunststoffscheibe montiert (siehe Abbildung A.9), welche über Schlitze in der äußeren Phase verfügt. Diese Schlitze bewegen sich durch die am Encoder befindlichen optischen Messsysteme welche bei ihrer Bewegung die beschriebenen Quadratursignale erzeugen. Hieraus ergibt sich folgende Gleichung für die Auflösung

$$Z = \frac{360\,\mathrm{deg}}{4 \cdot n_{\mathrm{Enc}}} = 2{,}5\frac{\mathrm{deg}}{Impuls}. \tag{3.3}$$

Der Energieverbrauch der Encoder beläuft sich bei 5V Betriebsspannung zu 11,6mA [36, S.1]. Zur Konzipierung und Herstellung einer Aufnahme des Antriebssystems (siehe Abbildung 3.2) stellen sich verschiedene Möglichkeiten. Um eine Flexibilität im Entwicklungsprozess zu gewährleisten wurde die Aufnahme mittels dem FDM-3D-Druckverfahren hergestellt. Bei dem verwendeten Filament handelt es sich um einen PLA-Kunststoff, welcher für die geringe Nutzlast eine Ausreichende Biegefestigkeit von 83Mpa bietet. Auf eine Optimierung des Flächenträgheitsmoments wurde bei der Entwicklung der Aufnahme auf Grund der geringen Nutzlast verzichtet. Eine Verrundung der Kanten an Strukturübergängen mit einem Winkel $\leq 90°$ wurde implementiert um Spannungsspitzen zu minimieren. Bei der Entwicklung der Aufnahme wurde die finale Aufbauhöhe des Chassis mitberücksichtigt um zu gewährleisten, dass genügend Abstand zum Boden besteht, um die später verbaute Elektronik implementieren zu können.

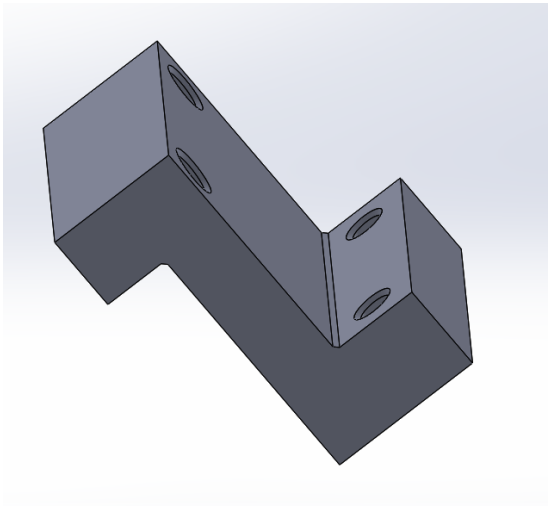


Abbildung 3.2: CAD-Modell der Aufnahme des Antriebssystems

Die Bohrungen zur Montage der Aufnahme wurden nach den Vorgaben für DIN912 Schrauben erstellt und sind für die Gewindegröße M5 ausgelegt. Zur Umsetzung des passiven Rades wie in Abbildung 3.1b gezeigt wurde eine Lenkrolle verwendet, welche mittels einer Aufnahme am Chassis befestigt wurde. Beim Design der Aufnahme wurde berücksichtigt, das das Chassis parallel zum Boden ausgerichtet ist.

3.2.2 Chassis

Für die Konstruktion und Entwicklung des Chassis wurden die selben Designanforderungen berücksichtigt wie bei der Aufnahme des Antriebssystems. Eine hohe Flexibilität im Design ermöglicht es nachträgliche Änderungen und Ergänzungen am Chassis vorzunehmen. Aus diesem Grund wurde vom DLR vorgegeben das Chassis aus ITEM-Profiltechnik herzustellen. Bei ITEM-Profilen handelt es sich um extrudierte Aluminiumprofile, welche mittels diverser ITEM-

Verbindungstechniken montiert werden können. Die verwendeten Profile weisen einen quadratischen Querschnitt mit einer Kantenlänge von 40mm auf (siehe Abbildung 3.3)

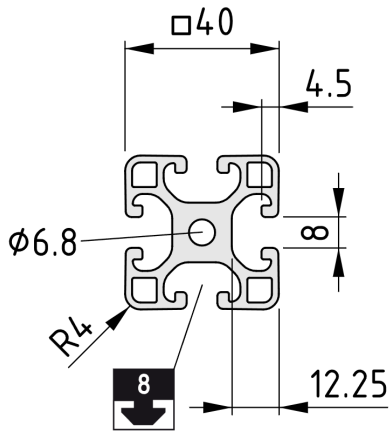
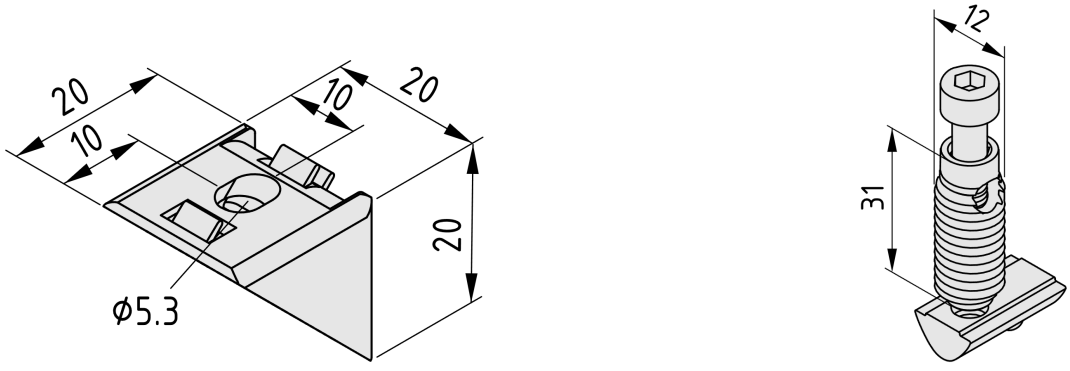


Abbildung 3.3: Querschnitt des verwendeten ITEM-Profils [17]

Der Unterbau zur Montage der Elektronik wurde aus Platzgründen aus ITEM-Profilen mit einer Kantenlänge von 20mm konstruiert. Um die Verbindungen zwischen den 40mm ITEM-Profilen, volumeneffizient zu gestalten wurden diese mittels des Automatik Verbindungssatzes von ITEM und den dazugehörigen Nutsteinen realisiert (siehe Abbildung 3.4b). Die ITEM-Profile des Unterbaus wurden mittels Verzinkter Winkel verbunden (siehe Abbildung 3.4a).



(a) Verzinkter ITEM-Winkel

(b) ITEM-Automatik-Verbindungssatz

Abbildung 3.4: Abbildungen der Verwendeten ITEM-Verbindungstechnik [17]

Unter Berücksichtigung der durch das DLR gestellten Randbedingung, dass das Testbett innerhalb einer Büroumgebung manövrierfähig sein muss, sowie über das nötige Volumen verfügen muss um Sensorik und Aktuatorik zu beherbergen, wurde für das Chassis eine rechteckige Form gewählt. Ein weiterer Grund hierfür sind die Beschränkungen in den Verbindungen welche mittels der ITEM-Profile möglich sind. Generell waren mit den vor Ort verfügbaren Arbeitsmate-

rialien ausschließlich 90° Verbindungen möglich. Aus diesem Grund wurde für das Chassis im Designprozess eine Kombination aus Rechtecksvolumen gewählt (siehe Abbildung 3.5).

Zur Erhöhung des Flächenträgheitsmoments, sowie zur Montage der Aufnahme der Lenkrolle wurden Querstreben vorgesehen, welche ebenfalls mittels des Automatik Verbindungssatzes (siehe Abbildung 3.4b) montiert wurden.

(a) Draufsicht des Chassis **(b)** Seitenansicht des Chassis

Abbildung 3.5: Chassis Ansichten

Des Weiteren wurden (siehe Abbildung 3.5b) Querstreben im Unterbau vorgesehen. Diese dienen als Montageschienen um im späteren Verlauf die elektronischen Komponenten zu montieren.

3.2.3 Hardware zur zentralen Datenverarbeitung

In der Auswahl der Hardware zur zentralen Datenverarbeitung sind anhand der vom DLR gestellten Randbedingungen zusätzliche Punkte zu beachten welche sich aus diesen ableiten.

Das Tesbett soll in der Lage sein die für „... die Lokalisierung und Annäherung notwendige Sensorik und Aktuatorik und die damit verbundene Datenverarbeitung innerhalb eines erweiterbaren Softwareframeworks zu verbinden“ (siehe Kapitel 3.1). Hieraus leitet sich ab, dass das System zur zentralen Datenverarbeitung einerseits über die nötige Rechenleistung verfügen muss um SLAM Algorithmik und Navigationsaufgaben lösen zu können. Des Weiteren sollte diese über eine API verfügen um eine Programmierschnittstelle zu ermöglichen. Zudem müssen Kommunikationsschnittstellen für Sensorik und Aktuatorik vorhanden sein. Aus der Randbedingung der Flexibilität leitet sich ab, dass das Softwareframework des Testbetts über einen größtmöglichen Funktionsumfang verfügt, um Limitierungen in der Softwareentwicklung zu minimieren. Konsequenz hieraus ist, dass die Hardware zur zentralen Datenverarbeitung in der Lage seien muss, generische Betriebssysteme wie Ubuntu oder andere auf Unix basierende Betriebssysteme betreiben zu können.

Aus diesen Abgeleiteten Randbedingungen stellen sich folgende eingebettete Systeme zur Auswahl.

Raspberry Pi 5

beim Raspberry Pi 5 handelt es sich um ein eingebettetes System der Raspberry Pi Foundation [41]. Der Raspberry Pi 5 verfügt über eine ARM64 quad-core CPU Architektur mit einer Taktrate von 2.4GHz pro Kern [41]. Des weiteren verfügt der Raspberry Pi 5 über eine dedizierte GPU zur Bilddatenverarbeitung [41]. Der Raspberry Pi 5 ist mit verschiedenen RAM größen verfügbar, hierbei stehen 4GB und 8GB zur Verfügung. Als Datenschnittstellen stehen neben dem PCIe Anschluss, zwei USB 2.0 und zwei USB 3.0 sowie eine Ethernet-Schnittstelle zur Verfügung [41]. Zudem Verfügt der Raspberry Pi 5 über einen 40 Pin GPIO Anschluss [41]. Da auf Grund der Projektkosten der Beschaffungspreis ebenfalls eine Rolle spielt wird dieser in der Auswahl mitberücksichtigt. Der Raspberry Pi 5 in der Konfiguration mit 8GB RAM ist zum Zeitpunkt dieser Abschlussarbeit zu einem Preis von 70€ erhältlich. Der Raspberry Pi 5 ist mit den nativen Raspian OS oder mit anderen Unix basierten Betriebssystemen betreibbar.

Raspberry Pi 4 B

beim Raspberry Pi 4 B handelt es sich um ein weiteres eingebettetes System der Raspberry Pi Foundation [40]. Der Raspberry Pi 4 verfügt über eine ARM64 quad-core CPU Architektur mit einer Taktrate von 1,8GHz pro Kern [40]. Des weiteren verfügt der Raspberry Pi 5 über eine dedizierte GPU zur Bilddatenverarbeitung [40]. Der Raspberry Pi 4 B ist mit verschiedenen RAM größen verfügbar, hierbei stehen 1GB, 2GB, 4GB und 8GB zur Verfügung. Als I/O's stehen, zwei USB 2.0 und zwei USB 3.0, 2 micro-HDMI, 2 MIPI Display-, 2 MIPI Kamera ports, Micro-SD Karten-Schnittstelle, sowie eine Ethernet-Schnittstelle zur Verfügung [40]. Zudem Verfügt der Raspberry Pi 4 B wie der Raspberry Pi 5 über einen 40 Pin GPIO Anschluss [40]. Der Raspberry Pi 4 B ist zum Zeitpunkt dieser Abschlussarbeit zu einem Preis von 40€ erhältlich. Der Raspberry Pi 4 B ist mit den nativen Raspian OS oder mit anderen Unix basierten Betriebssystemen betreibbar.

Nvidia Jetson AGX Origin

Der Nvidia Jetson AGX Origin wurde zur Lösung von komplexen Grafikproblemen konzipiert, weswegen dieser über eine sehr leistungsfähige GPU verfügt welche Taktraten von 930MHz bis 1.3GHz aufweist [32]. Der CPU ist in einer 8 Kern und 12 Kern Variante verfügbar und realisiert Taktraten von bis zu 2,2GHz [32]. Der Nvidia Jetson AGX Origin ist in zwei Varianten Verfügbar, mit 32GB DDR5 RAM und 64GB [32]. Zu den implementierten I/0's zählen 4 USB 2.0, 4 UART, 3 SPI, 4 I2S, 8 I2C, 2 CAN, sowie DMIC, DSPK und GPIOs [32]. Der Nvida Jetson AGX Origin ist mit dem Unix basierten Jetpack-Betriebssystem nutzbar [32]. Der Preis des Nvidia Jetson AGX Origin beläuft sich zum Zeitpunkt dieser Abschlussarbeit auf 1900€.

Nvidia Jetson Nano

Der Nvidia Jetson Nano stellt ein preisgünstigeres Modell des Nvidia Jetson Reihe da. Dieser verfügt ebenfalls über eine dedizierte GPU mit maximalen Taktraten von 921MHz [33]. Der Jetson Nano verfügt über eine Quad-Core ARM Architektur und erreicht maximale Taktraten

von 1,34GHz pro Kern [33]. Dieser Verfügt zu dem über 4GB DDR4 RAM sowie 16GB eingebetteten Speicherplatz [33]. Der Jetson Nano verfügt über folgende I/O's: 1 USB 3.0, 3 USB 2.0, 1 4-lane PCIe, 3 UART, 2 SPI, 4 I2C, 2 I2S, Ethernet sowie GPIO's. Des Weiteren ist zur Videodatenausgabe ein HDMI Anschluss vorhanden [33]. Der Jetson Nano ist zum Zeitpunkt der Abschlussarbeit für 290€ erhältlich.

Um eine Auswahl des eingebetteten Systems Anhand der zu Beginn des Kapitels 3.2.3 gennanten Randbedingungnen zu treffen, ist es notwendig diese mit den technischen Spezifikationen der genannten eingebetteten Systeme abzugleichen. Alle der genannten Systeme verfügen theoretisch über die notwendige Rechenleistung SLAM-Algorithmik zu betreiben und Navigationsaufgaben lösen zu können. Des Weiteren besitzen alle genannten Systeme Kommunikationsschnittstellen um Sensorik und Aktuatorik betreiben zu können. Die Produkte der Nvidia Jetson Reihe verfügen über ein größeres Repertoire an möglichen Kommunikationsschnittstellen. Da die Kommunikation von Aktuatorik und Sensorik über USB Schnittstellen stattfinden wird, ist dieser Punkt nebensächlich, auch wenn es die Möglichkeiten für Kommunikationsanbindungen für zukünftige zusätzliche Sensorik vereinfacht. Ein treibender Punkt zur Auswahl des eingebetteten Systems wird somit von den Anschaffungskosten, sowie den damit verbundenen Vorteilen bestimmt. Der Raspberry Pi 5 bietet hierbei bei geringen Mehrkosten zum Raspberry Pi 4 eine substanzielle Steigerung der CPU-Leistung sowie eine integrierte GPU. Die zusätzlichen Vorteile in Kommunikationsschnittstellen, CPU- und GPU-Leistung durch die Nvidia Jetson Produkte rechtfertigen zu diesem Zeitpunkt nicht die preislichen Mehrkosten dieser. Aus diesen Gründen wurde im Entwicklungsprozess der Raspberry Pi 5 in der 8GB RAM Variante als Hardware zur zentralen Datenverarbeitung ausgewählt.

3.2.4 Motorsteureung

Zur Konzipierung einer Motorsteuerung für die vom DLR vorgegebenen Motoren (siehe Kapitel 3.2.1) ist es notwendig, deren Eigenschaften sowie Betriebsmodi zu klassifizieren. Bei den vom DLR vorgegeben Motoren handelt es sich um Gleichstrom Bürstenmotoren, welche eine Betriebsspannung von 12V aufweisen [36]. Eine Möglichkeit der Steuerung von Gleichstrom Bürstenmotoren ist die Pulsweitenmodulation [9, S.415]. Auf Grund der Randbedingung der hohen Flexibilität wurde für das Konzept der Motorsteuerung jeweils eine eigene Subkomponente vorgesehen. Der zu verwendende Motortreiber wurde vom DLR als Vorgabe gestellt. Bei dem Motortreiber handelt es sich um einen zwei kanaligen H-Brücken basierten Motortreiber, welcher mittels PWM-Eingangssignalen eines Motorcontroller die Motorgeschwindigkeit steuern kann. Der Motortreiber ist für Ströme bis 30A ausgelegt und befindet sich somit im Rahmen des Leistungsbugets der Parallax Motoren [36].

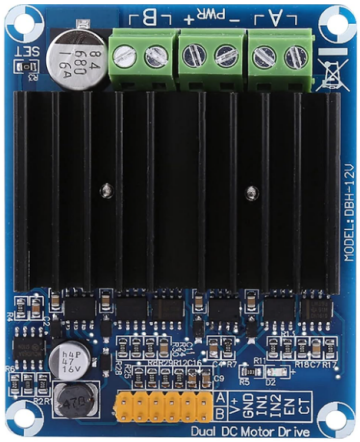


Abbildung 3.6: Abbildung des verwendeten Motortreibers in doppel H-Brücken Ausführung

Die grünen Schraubkontakte (siehe Abbildung 3.6) stellen die Ausgangsversorgungskontakte für Motor A und Motor B dar. Die in der Mitte liegenden Kontakte stellen die 12V Eingangsspannungskontakte zum betreiben der Motoren dar. Die Gelb eingefärbten Eingänge bezeichnen für IN1 und IN2 jeweils die PWM-Eingänge der Motoren A und B. Die Steuerung dieser folgt folgendem Schema:

Tabelle 3.1: Pin-Belegung und Betriebsmodi des Motortreibers

Motor	Vorwärts		Rückwärts	
	IN1	IN2	IN1	IN2
A	PWM	GROUND	GROUND	PWM
B	PWM	GROUND	GROUND	PWM

Die restlichen Eingänge bezeichnen Unterbrechungseingänge, sowie 5V Spannungsversorgungsausgänge und Ground Anschlüsse, welche jedoch nicht für die Anwendung in diesem spezifischen Fall relevant sind. Der Energieverbrauch des Motortreibers wurde mittels Labornetzteil zu 550mA bei 12V bestimmt. Zur Generierung der Regelung sowie der Steuerung der Motoren mittels PWM-Signalen soll zur Erfüllung der Anforderung nach hoher Flexibilität (siehe Kapitel 3.1) im Entwicklungsprozess ein unabhängiges System verwendet werden. Hierfür bieten sich im allgemeinen eine beliebige Anzahl an Microcontrollern an. Voraussetzung hierfür ist das diese über PWM-Ausgänge verfügen um den Motortreiber zu steuern, sowie einen seriellen Ausgang besitzen um eine Anbindung an die Hardware zur zentralen Datenverarbeitung zu schaffen. Im folgenden wird eine Auswahl einiger Microcontroller vorgestellt.

Arduino Nano

Der Arduino Nano stellt ein Microcontroller, welcher auf der Chipplatform ATmega328 basiert dar [3, S.2]. Der ATmega328 ist ein 8-bit Prozessor mit niedrigem Stromverbrauch, welcher Taktraten von bis zu 16MHz realisieren kann [3, S.2]. Der Microcontroller verfügt über 32kB internen Speicher und 2kB SRAM und einem separaten Oszillator zum einhalten der Taktzeiten [3, S.2]. Des Weiteren verfügt dieser über 20 digitale I/O's, 8 Analoge, 6 PWM Ausgänge sowie eine SPI Schnittstelle [3]. Der Energieverbrauch des Arduino Nanos beläuft sich auf 19mA bei 5V unter Maximallast [3]. Der Arduino Nano weist einen Betriebstemperaturbereich von 85°C bis -20°C auf. Der Microcontroller lässt sich über C++ programmieren. Zum Zeitpunkt dieser Abschlussarbeit ist der Microcontroller zu einem Preis von 15€ erhältlich, baugleiche Microcontroller (Arduino Klone) sind bereits für um die 3€ erhältlich

Teensy 40

Der Teensy 40 der Firma PJRC basiert auf der Chipplatform ARM Cortex-M7 welcher eine Taktgeschwindigkeit von 600MHz in einer Auflösung von 64 bit aufweist [48]. Der Teensy 40 verfügt über 1024kB RAM, sowie einem Sockel für micro SD-Karten [48]. Der Teensy 40 verfügt über 40 digitale I/O's von denen 31 als PWM Ausgänge und 14 als analog Eingänge benutzt werden können [48]. Des Weiteren verfügt der Microcontroller über 7 serielle, 3 SPI, 3 I2C sowie 3 CAN Bus Schnittstellen [48]. Der Teensy 40 weist einen Energieverbrauch von 100mA bei 5V auf bei maximalen Taktraten [48]. Der Teensy 40 lässt sich mittels C/C++ programmieren. Zum Zeitpunkt dieser Abschlussarbeit ist dieser zu einem Preis von 30€ erhältlich.

Raspberry pi pico

Der Raspberry pi Pico stellt einen Microcontroller der Raspberry Pi Foundation dar. Dieser verfügt über einen zwei Kern cortex M0+ Prozessor welcher Taktraten von bis zu 133MHz realisieren kann. Der Microcontroller verfügt über 264kB SRAM, sowie einen Micro-USB Anschluss als Programmierschnittstelle [42, S.3]. Des Weiteren befinden sich 40 I/O's auf dem Microcontroller, von denen 26 GPIO's sind (welche auf 3,3V Logikspannung betrieben werden), von diesen sind 23 digitale GPIOS und 3 analog/digital [42, S.3]. Als Kommunikationsschnittstellen sind 2 UART, 2 I2C, 2 SPI sowie 16 PWM Kanäle verfügbar [42, S.4]. Der Raspberry pi pico lässt sich mittels Micropython und C/C++ programmieren. Zum Zeitpunkt dieser Abschlussarbeit ist der Raspberry pi pico zu einem Preis von 5€ erhältlich.

Jeder der oben genannten Microcontroller verfügt über die Möglichkeit über Ausgangskanäle PWM-Signale auszugeben, somit kommen zur Steuerung des Motortreibes prinzipiell jeder der oben genannten Controller in Frage. Auf Grund der geringen Kosten des Arduino Nanos sowie der ausreichend hohen Taktfrequenz des Prozessors zur Realisierung von Echtzeitausgaben, wie dem steuern des Motortreibers, fällt die Auswahl des Microcontrollers zur Umsetzung des Motorcontrollers auf den Arduino Nano.

3.2.5 Sensorik

Zur Auswahl der Sensorik sind die Randbedingungen sowie die Grundvoraussetzungen aus Kapitel 3.1 relevant, um eine voranstehende Auswahl über die in Kapitel 2.2 genannte Sensorik zu treffen. Aus Randbedingung 1 ist abzuleiten, dass die Sensorik in der Lage sein muss Informationen über die Position des Testbetts im zweidimensionalen Raum zu generieren. Aus Randbedingung 3 lässt sich ableiten, dass die Implementierte Sensorik Daten zur Kartographierung des Büroraums bereitstellen können muss. Zuletzt stellt die Grundvoraussetzung 2 in Kombination mit Randbedingung 2 die Bedingung, dass die implementierte Sensorik in der Lage sein muss mit einer genügenden Auflösung den Nah- und Fernbereich zu erfassen um eine Navigation innerhalb diesem zu ermöglichen. Die nachfolgende tabellarische Aufstellung der Sensorik dient dem Vergleich dieser Aspekte um eine Vorauswahl der zu implementierenden Sensorik zu treffen. Hierbei wird die Anwendbarkeit der Sensorik im Nah und Fernbereich zur Positionsermittlung, die Möglichkeit des kartographierens Anhand der Sensordaten, sowie die Anwendbarkeit in einer Büroumgebung gegenübergestellt.

Tabelle 3.2: Sensorvergleich

Sensorart	Nahbereich	Fernbereich	Kartographiern	Anwend. in Büroumgebung
LIDAR	Ja	Ja	Ja	Ja
Stereo-Kamera	Ja	Nein	Ja	Ja
GNSS	Nein	Ja	Nein	bedingt
RADAR	Nein	Ja	Ja	bedingt
Startracker	Nein	Ja	Nein	Nein
Sonnensensor	Nein	Ja	Nein	Nein
Magnetometer	Nein	Ja	Nein	Nein
IMU	bedingt	bedingt	Nein	Ja
Autoident-Sys.	ja	Nein	Nein	Ja

Im Folgenden soll zusätzlich auf die Klassifizierung „bedingt“ aus Tabelle 3.2 eingegangen werden:

GNSS

GNSS Sensoren eignen sich wie in der Klassifizierung dargestellt nur bedingt innerhalb einer Büroumgebung zur Lokalisierung (siehe Kapitel 2.2.4). Aufgrund der Materialeigenschaften der tragenden Struktur des Bürogebäudes und der Wechselwirkung des GNSS-Signals mit dieser verschlechtert sich die Auflösung einer Positionsbestimmung via GPS in einem unvorhersehbaren Ausmaß. Somit ist das GNSS-System nicht auf die Größenordnung der in Randbedingung 3 genannten Genauigkeit skalierbar, auch wenn eine ungefähre Lokalisierung mittel GNSS möglich wäre.

RADAR

RADAR-Sensorik eignet sich nur bedingt zum Einsatz innerhalb einer Büroumgebung und ist stark von der Auslegung des RARAR-Systems Abhängig. Der in Kapitel 2.2.2 erwähnte „Clutter-

Effekt“ stellt in den begrenzten Abmaßen einer Büroumgebung einen zu hohen Einfluss auf die Daten welche durch eine Messung mittels RADAR-Sensorik generiert werden können dar. Ob eine Anpassung der Wellenlänge diese Effekte minimieren würde bleibt hierbei offen und nicht Teil dieser Abschlussarbeit.

IMU'S

Die Klassifizierung von IMU's als bedingt einsetzbar zur Lokalisierung im Nah- und Fern-Bereich resultiert aus den in Kapitel 2.2.8 genannten Addition der mittleren quadratischen Abweichungen der Positionsmessung mittels IMU-Sensorik. Somit ist die Qualität der ermittelten Daten stark Abhängig von dem Messzeitraum, sowie von der Sensorikkombination welche innerhalb des Testbetts gewählt wird.

Aus der Tabelle 3.2 ergibt sich, dass zwei Sensorarten in Kombination alle Voraussetzungen zur Lokalisierung im Nah-, Fern-Bereich in der Kartographierung und in der Anwendbarkeit innerhalb einer Büroumgebung erfüllen. Bei dieser Sensorik handelt es sich um LIDAR-Sensoren und Stereo-Kameras. Im folgenden wird nur eine Auswahl für Stereo-Kameras vorgestellt, da das DLR für dieses Projekt einen spezifischen LIDAR-Sensor gestellt hat.

OKDO LD06 LIDAR

Der OKDO LD06 LIDAR Sensor stellt einen als Raspberry pi Erweiterung konzipierten LIDAR Sensor dar. Der LIDAR-Sensor verwendet die in Kapitel 2.1.1 beschriebene TDOA-Methode zur Entfernungsbestimmung [15, S.1]. Das LIDAR-System arbeitet mit einer Messfrequenz von 4500Hz und weist einen 360° Messbereich ohne toten Bereich auf, siehe hierfür 2.12a [15, S.1]. Das LIDAR-System verwendet ein UART Interface zur Stromversorgung sowie dem Übermitteln der Daten [15, S.2]. Ein UART zu USB 2.0 Breakout-Board wurde verwendet um die Abstandssignale des OKDO LD06 LIDARS an die Hardware zur zentralen Datenverarbeitung (siehe Kapitel 3.2.3) weiter zu geben. Die Baudrate der Signalübertragung beträgt 230400 [15, S.2]. Die Eingangsspannung des LDIAR-Sensors beträgt 5V, die Logikspannung beträgt 3,3V. Die integrierte Laserdiode arbeitet bei einer Wellenlänge von 905nm bei 25mW und somit im Infrarotbereich [15, S.2]. Der Arbeitsbereich des LIDAR-Sensors zum Messung von Distanzen beläuft sich laut Datenblatt auf 0,02m bis 12m [15, S.3]. Die Genauigkeit des Sensors ergibt sich nach Herstellerangaben zu 30mm im Mittel [15, S.3]. das OKDO LD06 Lidar ist zum Zeitpunkt dieser Abschlussarbeit zu einem Preis von 45€ erhältlich.

ZED2 AI Stereo-Kamera

Die ZED2 AI Stereo-Kamera ist ein Kombinationssensor aus dem Hause Stereolabs, welcher eine Stereo-Kamera, sowie verschiedene Bewegungs- und Umweltsensoren mit einander Kombiniert [46, S.2]. Die ZED2 ist in der Lage dynamische Auflösungen in der Kamera zu realisieren. Diese Auflösungen ergeben sich zu 2208x1242 bei 15 FPS, 1920x1080 bei 30 FPS, 1280x720 bei 60 FPS und 672x376 bei 100 FPS [46, S.2]. Das FOV der Kameralinsen ergibt sich zu 110° horizontal, 70° vertikal und 120° diagonal [46, S.2]. Die aus der Stereo-Kamera generierten Tiefeninformationen

können laut Datenblatt in einem Bereich von 0,3m bis 20m realisiert werden und weisen eine Genauigkeit von weniger als 1% bei 0,3m und weniger als 5% bei 15m angegeben [46, S.2]. Der I/O der Stereo-Kamera stellt sich in Form eines USB-C Anschlusses [46, S.2]. Zu dem verfügt die ZED2 über folgende zusätzlich Sensorik: Ein Barometer, Temperatursensor, Gyroskop, Magnetometer, sowie einen Beschleunigungssensor [46, S.2]. Der Energieverbrauch der ZED2 beträgt 380mA bei 5V [46, S.2]. Die ZED2 kann mittels verschiedener Betriebssysteme betrieben werden, hierzu zählen Windows 10, Debian-Systeme wie Ubuntu, sowie dem Jetson L4T [46, S.3]. Die Mindestsystemanforderung sind ein Zweikern CPU mit 2,4GHz, 4GB RAM sowie ein 2GB GPU [46, S.3]. Die ZED2 AI Stereo-Kamera ist zum Zeitpunkt dieser Abschlussarbeit zu einem Preis von 565€ erhältlich.

OAK-D Lite

Die OAK-D Lite Stereo-Kamera verfügt über eine zentrale Farbkamera mit einer Auflösung von 4208x3120 bei 60 FPS, sowie einer automatischen Fokusfunktion [22]. Das FOV des Kamerasystems ergibt sich zu 73° horizontal, 58° vertikal und 86° diagonal [22]. Der Maximale Energieverbrauch der OAK-D ergibt sich zu 6W bei 5V [22]. Das Stereo-Kamera-system wird über zwei Graustufenkameras umgesetzt welche über eine Auflösung von 640x480 bei 120 FPS verfügen [22]. Das Graustufenkamera-System ist mit der zentralen Farbkamera gekoppelt um Texturinformationen aus diesen kombinierten Bildern zu generieren. Dieser Können zu Rekonstruktion sowie zum Abgleich von vorher implementierten Modellen verwendet werden, um diese im Bereich der Objekterkennung anzuwenden [22]. Die Tiefeninformationen können in einem Bereich von 0,4m bis 8m generiert werden. Die Genauigkeiten der Tiefeninformationen ergibt sich laut Datenblatt zu 2% im Bereich 0,4m - 3m, 4% bei 3m bis 6m und 6% bei 6m-8m [22]. Das Kamerasystem verfügt einen BMI270 6-Achsen IMU-Sensor sowie einem Magnetometer [22]. Des weiteren verfügt die OAK-D über interne Vorverarbeitungshardware zur Umsetzung von Objekterkennung sowie Neural Networking [22]. Die OAK-D Lite ist zur Zeit dieser Abschlussarbeit zu einem Preis von 150€ erhältlich.

Im Vergleich zwischen der OAK-D Lite und ZED2 AI lassen sich folgende Punkte vergleichen:

Tabelle 3.3: Vergleich der ZED2 AI und OAK-D Lite

Eigenschaft	OAK-D Lite	ZED2 AI
FOV	86°	120°
Auflösung	4k	2k
Reichweite max	8m	20m
Zusatzsensorik	IMU	IMU, Barometer, Temp. Sensor
Preis	150$	450$
min. Systemanforderungen	Nein	Ja

Im Hinblick auf die in Kapitel 3.1 genannten Randbedingungen und Grundvoraussetzungen erfüllen beide Systeme diese im vollen Umfang, die ZED2 AI weist einen minimal geringen Nahbereich auf, dieser ist laut Datenblatt bei 0,3m im Vergleich zu den 0,4m der OAK-D Lite. Dies stellt allerdings einen marginalen Unterschied dar. Die zu erfassenden Tiefeninformationen

werden in beiden Fällen mit einer ähnlichen Genauigkeit generiert, was zu eine ähnlichen Endauflösung führt. Auf Grund der Tatsache, dass Mindestsystemanforderungen zum Betrieb der ZED2 AI Stero-Kamera bestehen und der Preis dieser 300€ höher angesetzt ist, besteht keine ausreichende Rechtfertigung zur Auswahl dieser über der OAK-D Lite. Auch wenn die ZED2 AI über eine höhere Reichweite in der Tiefenmessung sowie mehr interne Umweltsensorik verfügt. Aus diesem Grund fällt die Auswahl des Stereo-Kamera-Sensors für das Testbett auf die OAK-D Lite.

3.2.6 Energieversorgung

Um eine Planung der Energieversorgung anzustellen, müssen die verschiedenen Spannungen berücksichtigt werden, welche zum Betrieb der in den Kapiteln 3.2.1, 3.2.3, 3.2.4 und 3.2.5 ausgewählten Komponenten benötigt werden. Alle genannten Komponenten, ausgenommen der Motortreiber, werden mit 5V betrieben. Der Motortreiber weist eine Arbeitsspannung von 12V auf, somit müssen die Spannungen 5V und 12V im System bereitgestellt werden. Um diese Voraussetzung zu gewährleisten müssen Spannungswandler eingeplant werden. Diese Spannungswandler wurden vom DLR gestellt. Bei diesen handelt es sich um DC/DC Spannungswandler der Firma Bauer welche einen Leistungsbedarf von 9,35mA bei 12V aufweisen. Um eine ausreichende Planung für die Energieversorgung aufzustellen muss das notwendige Leistungsbudget, welches sich aus der Summe der Einzelkomponenten ergibt, erfasst werden. Hierfür folgt nun eine Auflistung der elektrischen Komponenten sowie deren Leistungsbedarf:

Tabelle 3.4: Leistungsbedarf der verwendeten Komponenten

Komponente	Spannung	Strom	Anzahl	Watt
Raspberry pi 5	5V	5A	1	25W
Arduino Nano	5V	25mA	1	0,125W
Quad Encorder	5V	11,6mA	2	0,116W
Motortreiber	12V	550mA	1	6,6W
OKDO LD06 LIDAR	5V	180mA	1	0,9W
OAK-D Lite	5V	1,2A	1	6W
Parallax Motoren	12V	1,5A	2	36W
Spannungswandler	12V	9,35mA	2	0,2244W
gesamt	5V/12V	9,99A	11	75,08W

Die Leistung in Watt wurde nach Folgender Formel berechnet

$$P = IU. \tag{3.4}$$

Das Gesamtleistungsbudget der Energieversorgung ergibt sich somit zu 75,08W. Zur Stromversorgung des Systems wurde ein Lithium-Polymer-Akkumulator vorgesehen. Hierfür wurde eine vier-Zellen Konfiguration ausgewählt, die Nennspannung des Akkus beträgt somit 14,8V nomi-

nal. Der verwendete Akku besitzt eine Kapazität von 10Ah und ist somit in der Lage das System für 1h bei Volllast mit Strom zu versorgen. Die Berechnung hierfür ergibt sich wie folgt

$$t = \frac{C}{I} = \frac{10Ah}{9{,}99A} = 1{,}001h \approx 1h. \tag{3.5}$$

Auf Grund der Tatsache, dass weder die Motoren, noch der Raspberry pi 5 konstant auf voller Auslastung Betrieben werden, ergeben sich effektiv wesentlich längere Einsatzzeiten für den beschriebenen Akku.

Zu einer Erhöhung der Sicherheit im Betrieb des Roboters wurde des weiteren ein Not-Aus-Schalter sowie eine 20A Sicherung in der Stromversorgung eingeplant. Die Schaltung der Energieversorgung und Datenverbindung ergibt sich vereinfacht zu (siehe Abbildung 3.7):

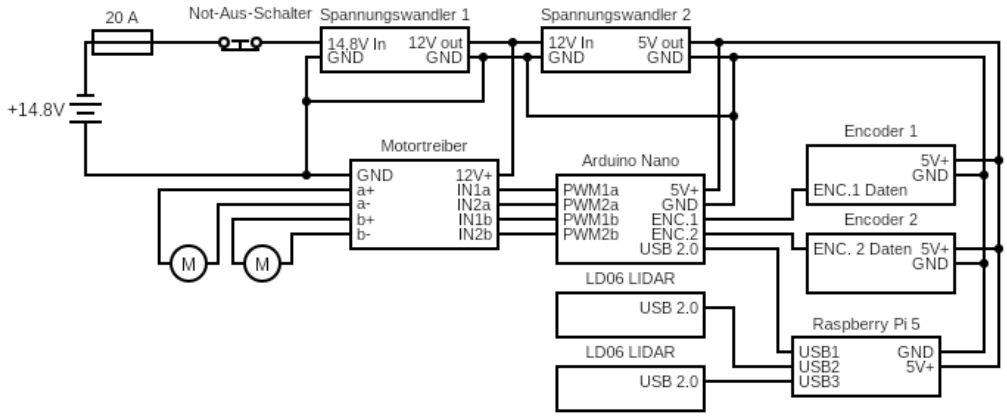


Abbildung 3.7: Abbildung des Schaltplans des Testbetts

3.3 Benchmarking der Sensorik

Zur Benchmarking der Sensorik sollen die arithmetischen Mittel bei verschiedenen Distanzen, sowie deren Varianz bestimmt werden. Hierfür werden die in ROS2 generierten Sensordaten des LIDAR-Systems verwendet. Zur Einordnung, sowie zum Bezug auf die Grundvoraussetzung 2, werden die Distanzen 7,5m, 4m, 1m, 0,65m, 0,3m, 0,15m, 0,05m verwendet. Das zur Auswertung verwendete Python-Skript kann dem Anhang B.1 entnommen werden. Die jeweiligen Messungen werden werden in einem Zeitraum von 10s Aufgenommen. Zur Analyse des Messdaten, ist es notwendig die Anzahl an Messpunkten pro Umdrehung des Sensors zu bestimmen. Hierfür werden die in Kapitel 3.4 dargelegten Werte aus der Kopfzeile der .csv-Datei verwendet. Hierbei

wird der Bereich zwischen den Variablen „anlge_min“ bis „angle_max“ durch die Größe des Winkelinkrements „angle_increment“

$$n = \frac{6{,}2831 Rad}{0{,}0139 Rad} = 452 \tag{3.6}$$

dividiert.

Nach Kontrolle der Daten sind pro Umdrehung nur 128 Entfernungsdatenpunkt vorhanden. Grund hierfür können fehlerhafte Zahlenwerte innerhalb des vom Hersteller bereit gestellten Hardwaretreibers des LIDAR-Sensors sein.

Anhand der 128 vorhandenen Entfernungsdatenpunkte ergibt sich das Winkelinkrement zu

$$w = \frac{360\,\mathrm{deg}}{128} = 2{,}81\,\mathrm{deg}\,. \tag{3.7}$$

Um den Startpunkt der Distanzmessung des LIDAR-Sensors zu ermitteln und um auf systematische Messfehler zu überprüfen, wird eine Initialmessung durchgeführt. Bei dieser Initialmessung wird das Chassis des LIDAR-Sensors abgeklebt um den Nullpunkt der Messung zu definieren. Nach einer 10s Messung werden die von ROS2 generierten Daten als .csv-Datei exportiert und mittels Python (siehe Anhang B.1) ausgewertet. Zur Analyse der Daten werden die Mittelwerte der Spalten 10 bis 138 gebildet. Diese Werte stellen die Mittelwerte der winkelspezifischen Messdaten über das Zeitintervall von 10s dar. Im Anschluss hierzu wird der Mittelwert dieser Mittelwerte, sowie ihre Standardabweichung gebildet (siehe Anhang B.1). Die generierten Daten ergeben sich hierbei zu:

Tabelle 3.5: Mittelwert und Standardabweichung der Nullmessung über 10 Sekunden

Mittelwert	Standardabweichung
0,0028m	0,00203m

Auffällig ist, dass die Standardabweichung in der Größenordnung des Mittelwerts liegt, was für eine starke Streuung der Messwerte im Bereich der Nullmessung spricht. Wie sich die Länge der Distanzmessung auf die Standardabweichung und die Mittelwerte auswirkt wird im folgenden untersucht.

Zur Vermessung der Genauigkeit über verschiedene Distanzen wird eine Testbank aufgebaut (siehe Abbildung 3.8) in welcher die geradlinige Entfernung x zu einem Ziel bestimmt werden kann. Um die Zuordnungsbarkeit der generierten Sensordaten zu erhöhen wird der nicht in Richtung des Ziels zeigende Sensorbereich analog zur Nullmessung abgeklebt. Dieser Bereich wird durch die Geraden a und b dargestellt (siehe Abbildung 3.8).

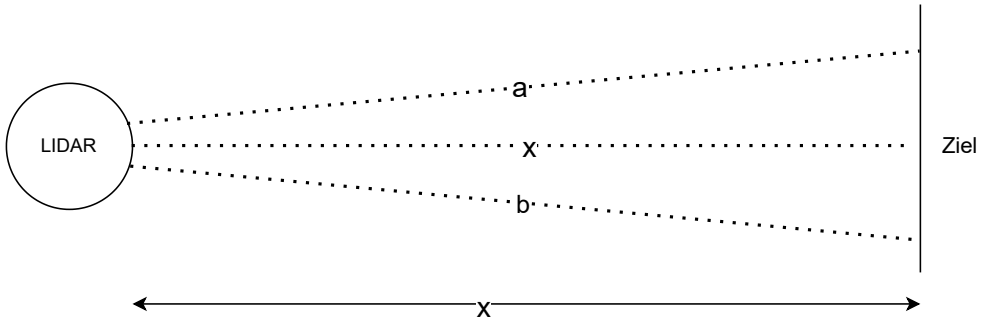


Abbildung 3.8: Abbildung der Testbank mit Geraden a und b sowie der Distanz x

Die bei dieser Messung generierten Daten, lassen sich in drei Gruppen unterteilen. Messdaten welche, von ihrer Größenordnung im Bereich der Nullmessung liegen, Messdaten welche im Bereich der zu erwartenden Distanzmessung liegen und Messdaten die hinter dem Ziel liegen. Um die Genauigkeit der Messung zu erhöhen müssen die Längenunterschiede der Geraden a und b zu x berücksichtigt werden. Über den Kosinussatz ergibt sich der Zusammenhang, am Beispiel der Geraden a zu

$$a = x + \Delta x \tag{3.8}$$

und

$$cos(\beta) = \frac{x}{a}. \tag{3.9}$$

Durch Umformen von Gleichung 3.9 und einsetzten in Gleichung 3.8 ergibt sich

$$\Delta x = x\frac{(1 - \cos\beta)}{\cos\beta}. \tag{3.10}$$

Bei der in Kapitel 3.1 definierten Grundvoraussetzung 2 festgelegten Maximaldistanz von 7,5m und dem in Gleichung 3.7 berechneten Winkelinkrement von 2,81° ergibt sich die Abweichung

$$\Delta x = 7{,}5m\frac{(1 - \cos(2{,}81\,\mathrm{deg}))}{\cos(2{,}81\,\mathrm{deg})} = 0{,}00902m. \tag{3.11}$$

bei einer Abweichung von einem Winkelinkrement bei der Distanz 7,5m beträgt der Längenunterschied bereits $\approx$ 1cm. Um diesen Messfehler zu verringern, wird im Pythoncode zur Auswertung der Messdaten (siehe Anhang B.1) innerhalb jeder Messung das lokale Minimum der Distanzmes-

sung bestimmt. Da der lokale Minimalwert die kürzeste Distanz zwischen dem LIDAR-Sensor und dem Ziel darstellt. Dieser Messwert ist der am nächsten an der tatsächlichen Distanz x liegende Wert. Vor der Bestimmung des Minimalwerts werden die Daten ebenfalls gefiltert um Messungen zu exkludieren, welche das Ziel verfehlt haben. Dieser Filterbereich wird für jede Distanzmessung spezifisch angepasst und kann Tabelle 3.6 entnommen werden. Die sich aus diesem Aufbau und Selektion der Daten ergebende Werte sind in Tabelle 3.6 dargestellt.

Tabelle 3.6: Varianzen und Mittelwerte des LIDAR-Sensors bei verschiedenen Distanzen

Distanz	Mittelwert	Varianz	Filterbereich
7,5m	7,492m	0,0284m	7,3m - 7,9m
4m	4,000m	0,007m	3,8m - 4,3m
1m	0,997m	0,002m	0,8m - 1,3m
0,65m	0,651m	0,005m	0,4m - 0,8m
0,3m	0,320m	0,004m	0,2m - 0,5m
0,15m	0,157m	0,0000027m	0,06m - 0,2m
0,05m	0,06m	0,001m	0,03m - 0,1m

Die in der Tabelle 3.6 gezeigten Daten weisen eine Varianz auf welche sich über die verschiedenen Messdistanzen in der selben Größenordnung befinden. Durch die bestimmte Mindestabweichung von 1mm stärkt sich die Vermutung, dass die Datenqualität des implementierte LIDAR Sensors ausreichend genau für den Nahbereich ist.

Anzumerken ist, dass die Varianz der Nullmessung in der selben Größenordnung wie die Varianzen der restlichen Messungen liegt. Dies spricht zusätzlich für eine hohe Güte der erfassten Daten, zu dem liegen die Varianzen unterhalb der Herstellerangaben.

Die OAK-D Lite Stereo-Kamera wird im folgenden nicht weiter betrachtet, da der Sensor bei Bestellung im Transport verloren gegangen ist und eine nachträgliche Bestellung im Angesicht des Abgabetermins dieser Abschlussarbeit nicht mehr in Frage kommt. Die zu implementierenden Funktionalitäten des Testbetts, können jedoch auch ausschließlich mit der LIDAR-Sensorik abgebildet werden (was sich durch das Benchmarking der Sensorik bestätigt), auch wenn eventuell mit der Kombination von zwei Sensoren eine höhere Datenauflösung erzielt werden könnte.

3.4 Software des Testbettes

Als Lösung zur Integration der Hardware und Software wurde das im Rahmen des Standford-AI-Projects entwickelte Software Framework Robot Operating System 2 (ROS2) gewählt. Dieses Software Framework ermöglicht es eine Bandbreite an standardisierten Funktionen innerhalb des Testbetts zu integrieren, welche über verschiedene Robotikanwendungen elementar sind. Diese Funktionen können in Form von ROS2 nativen oder selbst erstellten Paketen abgerufen werden. Zu diesen Funktionen zählen Beispielsweise das integrieren von Aktuatorik und Sensorik, Darstellungs- und Simulationssoftware, Koordinatentransformationsabläufe in Abhängigkeit von vom Testbett übermittelten Daten, SLAM-Funktion, autonome Navigationsalgorithmik so-

wie Objekterkennungssoftwareschnittstellen. Das Software Framework wurde unter dem Aspekt der in Kapitel 3.1 genannten Grundvoraussetzung 1 gewählt, da dies die geforderte Flexibilität in der Entwicklung ermöglicht um Randbedingung 1-3 und Grundvoraussetzung 2 umzusetzen. Da es sich bei ROS2 um ein funktionell sehr umfangreiches Software-Framework handelt, werden in dieser Abschlussarbeit nur die Grundlagen dargestellt welche unmittelbar notwendig zur Umsetzung der im Testbett gezeigten Funktionen sind. Die Daten- und Funktionsstruktur welche ROS2 verwendeten werden in „Nodes“ und „Topics“ unterteilt [23]. „Nodes“ stellen hierbei die Ausführenden Softwaresegmente dar, welche eine spezifische Aufgabe lösen [23]. „Nodes“ können hierbei entweder „Topics“ abonnieren, um von diesen Daten zu erhalten, oder an diese veröffentlichen um eine Datenübermittlung zu realisieren [23]. Die Anzahl der „Topics“ an welche eine „Node“ abonniert oder veröffentlicht, ist hierbei beliebig groß [23]. „Nodes“ können beim Start mit Argumenten gefüllt werden, welche das Verhalten dieser definiert [23]. „Topics“ werden im ROS2 Framework zum übermitteln von Daten verwendet. Welches Format und welchen Umfang die Daten haben ist fest durch die „Topics“ bestimmt um einen standardisierten Datenaustausch zwischen „Nodes“ zu gewährleisten [23]. „Topics“ stellen standardisierte Nachrichtentypen bereit, welche von den „Nodes“ als Informationseingänge oder Ausgänge verwendet werden können [23]. „Nodes“ und „Topics“ können von verschiedenen Systemen auf denen ROS2 betrieben wird und welche sich im selben Netzwerk befinden automatisiert über den DDS gefunden werden um über das Netzwerk Daten auszutauschen [23].

Das Festlegen von Startargumenten für verschiedene „Nodes“, sowie das automatisierte Starten mehrerer „Nodes“, welche miteinander interagieren sollen, kann mittels eines Python-Skripts realisiert werden. Diese Skripte, welche den Start von verschiedenen „Nodes“ und das vorherige festlegen ihrer Argumente ermöglichen werden auch „launch-files“ genannt.

Als Vorlage für die Umsetzung des Testbetts wurde ein bereits Vorhandenes Github-Projekt [28] verwendet, welches im Laufe der Entwicklung des Testbetts angepasst wurde.

Grundlage der in dieser Abschlussarbeit darzustellenden Funktionalitäten ist das Robotermodell. Hierbei werden einerseits die Abmaße und Dimensionierungen der verschiedenen Komponenten erfasst, jedoch auch die Freiheitsgerade der beweglichen Teile des Testbetts. Diese Definition ist notwendig, damit anstehende Koordinatentransformationen, welche für die Darstellung der Kartographierung und der Visualisierung des Testbetts unabdingbar sind, durchgeführt werden können. Des Weiteren können innerhalb des Robotermodells die Trägheitsmomente der Komponenten festgelegt werden um das Testbett akkurater simulieren zu können. Das Robotermodell wird mittels XML-Syntax beschrieben und beinhaltet einerseits die Verknüpfungen zwischen verschiedenen Teilen, sowie deren Referenz zu einem Ursprungskoordinatensystem [34]. Die im Robotermodell beschrieben Komponenten werden einem jeweiligen Koordinatensystem zugeordnet, welche innerhalb von ROS2 als Frames bezeichnet werden [34]. Diese Frames sind über Gelenke („Joints“), welche gewisse Freiheisgrade aufweisen, verbunden. Hierbei wird die Definition von „Parent frames“ und „Child frames“ verwendet. Damit eine eindeutige Koordinatentransformation zwischen den „Parent frame“ und „Child frame“ erfolgen kann, darf ein „Child frame“

nur einen „Parent frame“ besitzen. Diese Zusammenhänge können ebenfalls der Abbildung 3.9 entnommen werden.

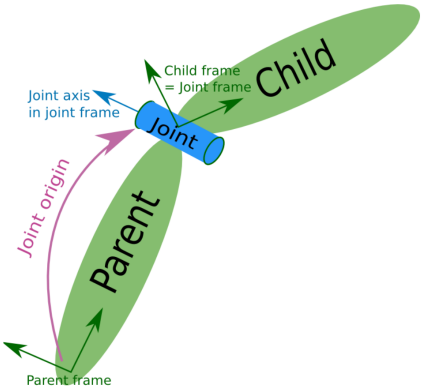


Abbildung 3.9: Abbildung der Koordinatensysteme und Gelenke im Robotermodell [34]

Zur Ansteuerung und Regelung der Motoren wird das ROS2 interne „ROS2_Control“-Paket mit dem dazugehörigen „Differential_Drive“ Plugin verwendet [34]. Diese ermöglicht es Roboter welche mit Differentialantrieben ausgestattet sind zu steuern. Die Steuerung erfolgt mittels des Nachrichtentyps „Twist“ [34]. Beim Nachrichtentyp „Twist“ handelt es sich um einen sechsstelligen Vektor. Drei der sechs Einträge enthalten einen abstrahierten Wert für die translatorischen Geschwindigkeitskomponenten zwischen -1 und 1. Die restlichen drei Einträge enthalten die rotatorischen Geschwindigkeitskomponenten welche als Zahlenwert zwischen -1 und 1 dargestellt werden [34]. Die Übermittlung dieser Werte erfolgt über einen Hardwaretreiber an den Arduino Nano, welcher diese Werte in geeignete PWM-Signale umwandelt um mit diesen die Motoren zu steuern. Für den C++ Code des Arduino Nanos wurde ein bereits bestehenden Github-Projekt verwendet und für den im Testbett verwendeten Motortreiber angepasst [10]. Der ROS2 Arduino Treiber wurde ebenfalls einem bestehenden Github-Projekt entnommen um mit diesem die serielle Kommunikation über ROS2 zu ermöglichen [29]. Die sich an den Rädern befindenden Encoder generieren, bei der Bewegung der Räder, Radpositionsdaten. Diese Daten werden vom Arduino Nano, anhand ihrer in Kapitel 3.2.1 definierten Auflösung, interpretieren und an das „ROS2_Control“-Paket als Rückkopplung zur Regelung des Antriebs gegeben. Die Übermittlung der Daten erfolgt über eine serielle USB-Schnittstelle zum Raspberry Pi 5. Die serielle Datenschnittstelle zur ROS2 Umgebung wurde mittels eines Opensource Treibers für serielle Eingabedaten für UNIX und Windowssysteme realisiert [51].

Zur manuellen Generierung der „Twist“-Nachrichten wurde ein Logitech f710 Gamepad verwendet. Zur Realisierung der Funktionalität dieses wurden UNIX native „Joystick“-Treiber verwendet, sowie das ROS2 native „teleop_twist_joy“-Paket [34]. Dieses Paket ermöglicht es Eingabetasten und Joystickpositionen auf dem Logitech f710 spezifischen Funktionen zuzuweisen und mit diesen „Twist“-Nachrichten zu erzeugen. Diese „Twist“-Nachrichten werden über Eingabeparameter an ein geeignetes Topic übergeben. Die „ROS2_control“-Node, welche über das

„ROS2_control“-Paket aufgerufen werden kann, ist an dieses „Topic“ abonniert um die „Twist“-Nachrichten vom Joystick zu erhalten [34].

Ein weiterer elementarer Teil, stellt das ROS2 native Paket „robot_state_publisher“ dar. Beim Abruf der „robot_state_publisher-Node“ wird zeitgleich ein „Topic“ mit dem Namen TF generiert. Das „Topic“ TF publiziert Koordinatentransformationen in Abhängigkeit der von der „ROS2_Control-Node“ erhaltenden Encoder-Daten bezüglich der Radpositionen [34]. Diese Transoformationen werden von der „robot_state_publisher-Node“ berechnet [34]. Als Grundlage dieser Berechnungen und deren Änderung über die Zeit dient das vorher definierte Robotermodell. Diese Daten können zur Odeometrie verwendet werden um eine Annäherung der relativen Position des Testbetts im Verhältnis zur Startposition zu generieren [34]. Diese Annäherung ist, ähnlich wie im Kapitel 2.2.8 sich zeitlich aufsummierenden Fehlern unterlegen. Der Grund hierfür ist das die Berechnungen nicht den Schlupf der Reifen Abbilden und die Encoder eine fest definierte Auflösung aufweisen (siehe Kapitel 3.2.1). Eine Visualisierung des Robotermodells, sowie der Koordinatensysteme der Komponenten über die ROS2 native Software „RVIZ2“ kann 3.10 entnommen werden.

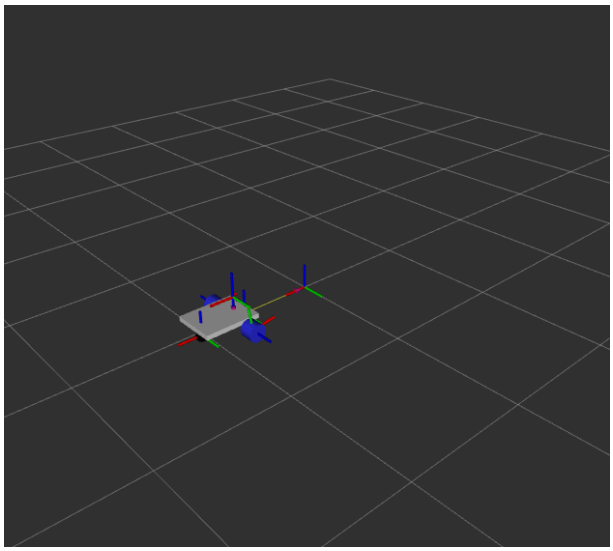


Abbildung 3.10: Abbildung der Koordinatensysteme und des Robotermodells in RVIZ2

Der Betrieb des OKDO LD06 LIDAR wurde mittels eines vom Hersteller bereitgestellten ROS2 Treibers durchgeführt [21]. Um die Funktionalität dieses Sensors sicherzustellen, müssen im „launch-file“ auf das zu veröffentlichende „Topic“ verwiesen werden, sowie den exakten USB Eingang am Raspberry Pi 5. Im Falle des Testbetts wurde die Zuweisung in Ubuntu nach dem Gerätenamen gewählt, sodass das LIDAR an jedem beliebigen USB-Anschluss angeschlossen sein kann. Das Topic an welchem die Daten des LIDAR-Sensors übermittelt werden trägt den Namen „Scan“. Zudem sind zusätzliche Parameter vorhanden, welche jedoch zur Umsetzung

dieser Abschlussarbeit keine Relevanz haben. Der Nachrichten-Typ welcher vom „Scan-Topic“ bereit gestellt wird beinhaltet folgende Informationen:

stamp

Dieser Eintrag stellt den Zeitstempel dar, an welchem Zeitpunkt die Messung aufgenommen wurde. Dieser Parameter ist an eine globale Zeittaktung im ROS2-System gekoppelt.

frame_id

Die „frame_id“ ist eine Referenz zum Robotermodell und legt fest welches Objekt im Robotermodell den LIDAR-Sensor darstellt.

angle_min

Der „angle_min“ stellt den Startwinkel einer Messung in Rad dar.

angle_max

Der „angle_max“ stellt den Endwinkel einer Messung in Rad dar.

angle_increment

Dieser Wert stellt die Größe der Winkelinkremente der Messung dar.

scan_time

„Scan_time“ stellt die Zeit, welche für den Scan benötigt wurde dar.

range_min

Der Wert „range_min“ erzeugt die minimal aufgenommene Distanz der Messung in Meter.

range_max

Dieser Eintrag erzeugt die maximale Distanz in der Messung in Meter.

ranges

Der Array „ranges“ beinhaltet alle Messungen welche innerhalb einer Umdrehung des LIDAR-Sensors aufgenommen wurden in Meter.

Die Messung dieser Werte kann über das ROS2 native Visualisierungstool „RVIZ2“ dargestellt werden. Innerhalb dieses Tools kann ebenfalls das Robotermodell, sowie die dazugehörigen Koordinatensysteme und ihre Änderung über die Zeit angezeigt werden wie in Abbildung 3.11 dargestellt.

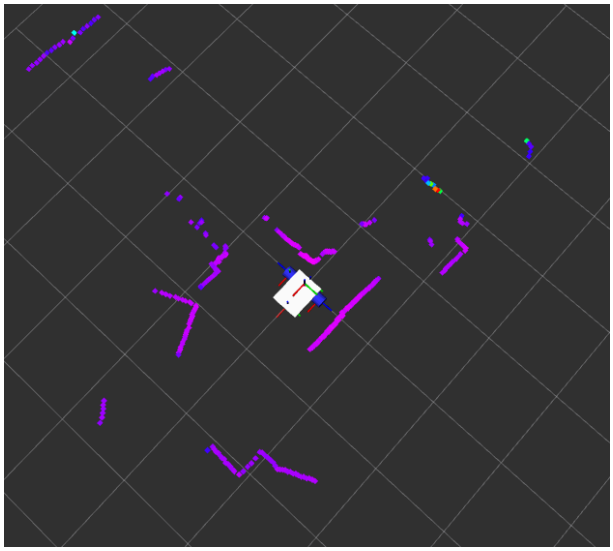


Abbildung 3.11: Abbildung des Robotermodells und der dynamischen LIDAR-Daten in RVIZ2

Zur Realisierung der SLAM Funktionalitäten, wurde das ROS2 native „SLAM_Toolbox“-Paket verwendet, welches bereits weit verbreitete Anwendung im Forschungs- und Industrie-Kontext findet [25]. SLAM-Methoden werden verwendet um Robotik-Anwendungen in Umgebungen mit stark variabler Größe eine genaue Positionierung und Lokalisierung innerhalb einer generierten Karte zu ermöglichen [25, S.1]. Der spezielle Anwendungsfall ergibt sich vor allem in Einsatzgebieten in denen Dienste wie GPS oder „Motion-Tracking“ keine genaue Positionierung und Lokalisierung ermöglichen können [25]. Die Grundlage dieser Positionierung und Lokalisierung sind Hauptsächlich LIDAR Sensoren, welche allerdings ebenfalls mit anderer Sensorik kombiniert werden kann [25, S.1]. Im Falle des Testbetts wurde der Operationsmodi des „Online asynchronen Mappings“ gewählt, da eine Echtzeitlokalisierung innerhalb der Karte am geeignetsten für Navigationsaufgaben innerhalb einer dynamischen Umgebung ist [25, S.4]. Das „SLAM_Toolbox“-Paket ist in der Lage in diesem Betriebsmodi dynamisch auf Änderungen in der Umgebung zu reagieren und die generierte Karte dementsprechend anzupassen [25, S.4].

Beim Start der SLAM-Toolbox-Node werden verschiedene „Topics“ erzeugt. Die „Node“ ist an das oben genannten „Scan-Topic“ und „TF-Topic“ abonniert und erhält von diesen Odeometriedaten, sowie die vom LIDAR aufgenommen Enfernungsmessungen [24]. Als Ausgabedaten veröffentlicht die „Node“ an ein „map-Topic“ innerhalb welchem auf die dynamische Karte zugegriffen werden kann, sowie ein „pose-Topic“, welches die berechnete Position im Kartenkoordinatensystem enthält [24]. Auf Grundlage des Kartenkoordinatensystems, kann eine Korrektur der durch die Odeometrie generierten Koordinatentransformationen stattfinden um eine Lokalisierung innerhalb der Karte zu ermöglichen [24]. Die Generierung der Karte kann 3.12 entnommen werden.

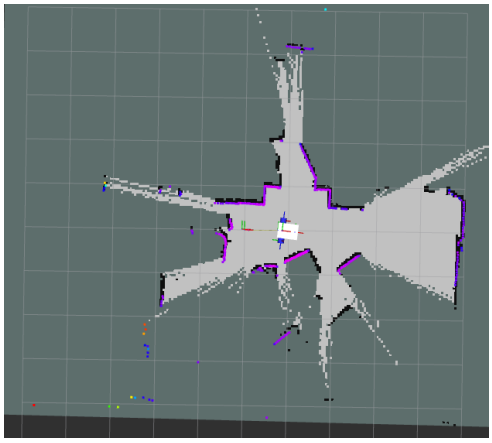


Abbildung 3.12: Abbildung der Kartengenerierung mittels der SLAM-Toolbox: Die hell graue Kolorierung stellt die erzeugte Karte dar, die farbige Kolorierung die LIDAR-Daten

Zur Implementierung der autonomen Navigation wird das ROS2 native Paket „NAV2“ verwendet. „NAV2“ stellt ein auf „Behavior Trees“ basierendes Ablaufprogramm dar [26, S.1]. Diese ist in der Lage auf dynamische Änderungen der Umgebung zu reagieren und innerhalb dieser zu navigieren [26, S.2]. „NAV2“ benutzt hierfür eine grobe probabilistische Repräsentation der Umgebung auf Grundlage von mittels SLAM-Funktionalitäten generierten Daten [26, S.2]. Zum Ausweichen von Hindernissen wird hierbei der DWA verwendet [26, S.2].

Die Architektur des „NAV2“-Pakets lässt sich in vier Typen einteilen,„Planner“, „Control“, „Costmap“ und „Recovery“. Die Aufgabe des „Planners“ innerhalb des „NAV2“-Paket ist es den kürzesten Weg zwischen aktueller Position und gesetztem Ziel zu kalkulieren [26, S.4]. Der „Control“-Typ besitzt die Aufgabe den besten lokalen Weg zum Ziel zu finden, sowie die für die Steuerung erforderlichen Signale zu generieren [26, S.4]. Hierfür werden lokale Informationen wie die vom LIDAR generierten „Scan“-Daten verwendet [26, S.4]. Die „Costmap“ (siehe Abbildung 3.13) stellt hierbei eine Überlagerung der durch „SLAM“-Funktionalitäten generierten Karte dar, welche die statistische Komplexität darstellt innerhalb dieser Karte lokal zu einem Ziel zu navigieren [26]. Ziele werden an das „NAV2“-Paket in Form von Vektoren übergeben, welche eine bestimmte Position und Ausrichtung innerhalb der Karte darstellen, an welche navigiert werden soll. Der „Recovery“-Typ wird verwendet um Navigationsfehler zu mitigieren [26, S.4]. Die Ablaufenden Modi im Falle eines Navigationsfehlers sind:

Clear Costmap

Dieser Modi löscht die Einträge der „Costmap“ im Falle eines Fehlers in der Wahrnehmung des Roboters und fängt an eine neue „Costmap“ zu generieren [26, S.4].

Spin

Hierbei wird ein Steuerungssignal erzeugt welches dazu führt, dass sich der Roboter im Kreis dreht, um freien Platz abzufahren und aus eventuellen lokalen Fehlern der Navigationsplanung zu manövrieren [26, S.4].

Wait

Der Roboter wartet an der nicht manövrierfähigen Position um eventuelle dynamische Hindernisse abzuwarten und oder um mehr Sensordaten zu sammeln [26, S.4].

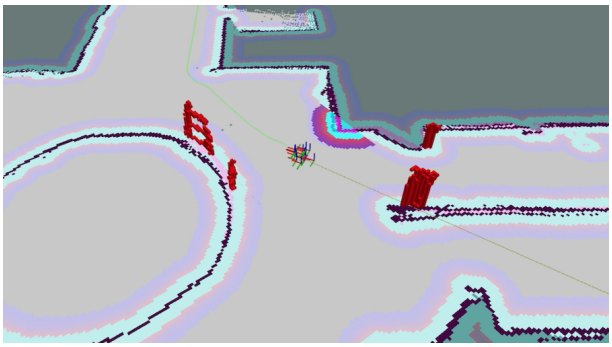


Abbildung 3.13: Abbildung einer Costmap in RVIZ2 als Überlagerung der SLAM-Toolbox Karte [26, S.5]

4 Autonome Navigation und Lokalisierung im Testbett

Dieses Kapitel beschäftigt sich mit der Analyse und Auswertung der durch das „SLAM-Toolbox“- und „NAV2“-Paket implementierten Funktionalitäten. Die Kartographierung, Lokalisierung und autonome Navigation des Testbetts wird hierbei in Referenz zu der verwendeten Sensorik bewertet. Zur Analyse der Kartographierungs- und Lokalisierungsfähigkeiten werden Ansätze analog zu Kapitel 3.3 verwendet. Die Analyse der Navigationsfähigkeiten erfolgt hierbei qualitativ.

Um eine Vergleichbarkeit der erhobenen Messdaten zu gewährleisten wird innerhalb der Büroumgebung (siehe Kapitel 3.1 unter Randbedingung 3) ein fester Parkour definiert. Dieser wird vom Testbett bei der Erhebung der Daten abgefahren. Ziel ist es mittels des Parkours alle möglichen Betriebsmodi abzudecken. Hierzu zählen Kurvenfahrten, Vorwärts- und Rückwärtsbewegungen, sowie Drehungen. Der ausgewählte Parkour kann Abbildung 4.1 entnommen werden.

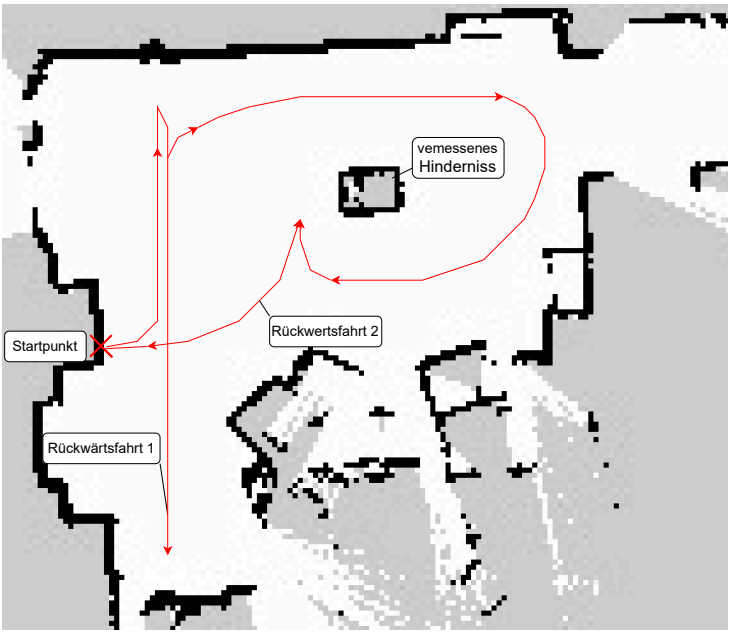


Abbildung 4.1: Abbildung des Parkours innerhalb der generierten Karte

4.1 Simultanes Lokalisieren und Kartographieren zur Positionsbestimmung

Die in Kapitel 3.4 beschriebene „SLAM-Toolbox“ exponiert beim Betrieb zwei „Topics“ (siehe Kapitel 3.4) welche für die Analyse der Lokalisierung und Karographierung von Bedeutung sind. Hierzu zählen das „map-Topic“ und das „pose-Topic“ [24]. Neben den von dem „SLAM-Toolbox“-Paket exponierten „Topics“ ist ebenfalls das „TF-Topic“ zur Analyse von Bedeutung. Das „map-Topic“ benutzt den Nachrichtentyp „OccupancyGridMessage“ welcher folgende Werte enthält [34]:

header

Der „header“ enthält eine Zählvariable der Nachrichtennummer, einen Zeitstempel an welchem die Nachricht aufgenommen wurde, sowie die „frame_id“ des Koordinatensystems in welchem die Karte veröffentlicht wird.

info

Der „info“-Teil der Nachricht enthält die Metadaten der Karte. Hierzu zählen die „map_loadtime“ (der Zeitpunkt an welchem die Kartendaten erstellt wurden), sowie „width“ und „hight“ welche die Pixelhöhe und -breite der Karte angeben. Der Wert „Resolution“ enthält die Auflösung in Meter pro Pixel. Ebenfalls wird in diesem Teil der Nachricht der Startpunkt der Generierung der Karte in Referenz zum Kartenkoordinatensystem festgehalten. Der letzte Eintrag des info-Teils stellt einen Array dar in welchem die Kartendaten gepeichert sind. In diesem werden die Wahrscheinlichkeiten (ausgedrückt als Zahl zwischen 0 und 100) festgehalten in welchen Pixeln sich Hindernisse befinden. Der Wert -1 drückt hierbei noch nicht ermittelte Daten aus und wird graphisch als Hellgrau dargestellt.

Das „pose-Topic“ kommuniziert mittels des Nachrichtentyps „PoseWithCovariance“ [34]. Diese Nachricht enthält folgende Einträge:

pose

Die „pose“ gibt einerseits die Position (in x-, y- und z-Koordinaten) des Roboters in Referenz zum Kartenkoordinatensystem in Metern an und andererseits die Ausrichtung des Roboters in Quaternionen.

covariance

Dieser Eintrag enthält die Kovarianzmatrix, welche die Dimensionen 6x6 aufweist. Die Kovarianzmatrix gibt die durch den Algorithmus estimierte Unsicherheit der sechs DOF's an, sowie ihre Abhängigkeit untereinander.

4.1.1 Daten der Kartographierung

Um eine Analyse der Kartographierfähigkeiten durchzuführen, ist es notwendig Vergleichsdaten zu generieren. Überprüft werden soll wie genau die generierte Karte die wirkliche (in Kapitel 3.1 unter Randbedingung 3) Büroumgebung darstellen kann.

Hierfür wird innerhalb einer Testumgebung ein Referenzpunkt in Form eines Hindernisses in der Büroumgebung implementiert, welche vorher vermessen wurde. Zudem werden die Distanzen zu den Wänden der Umgebung vermessen. Ebenfalls erfolgt eine Vermessung der charakteristischen Raumgrößen (siehe Abbildung 4.2). Anhand dieser Messungen kann die Genauigkeit der Karte quantitativ durch einen direkten Abgleich analysiert werden. Die Messung der Distanzen im Raum wurde mittels eines nach ISO16331-1 genormten Laserdistanzmessers analysiert. Der Laserdistanzmesser weist laut Datenblatt eine Messvarianz von ±1,5mm auf. Die durch das „SLAM_toolbox“-Paket generierte Karte kann aus dem „map-Topic“, oder direkt aus dem ROS2 nativen Programm „RVIZ2“ exportiert werden.

Bei dem implementierten Hindernis handelt es sich um eine Box mit den Abmaßen 0,37m x 0,54m. Da die Höhe der Box nicht für die Kartenbildung relevant ist. wurde diese nicht mit betrachtet. Die gemessenen Raumgrößen und Hindernisse können Abbildung 4.2 entnommen werden.

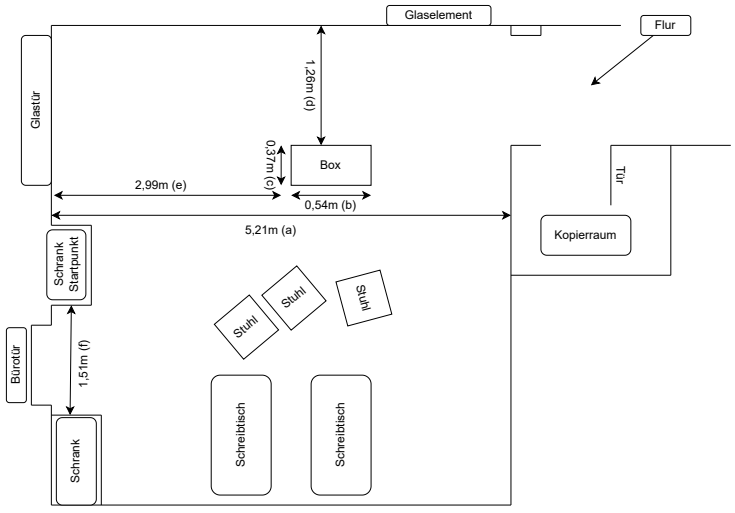


Abbildung 4.2: Abbildung der Raumgrößen der Büroumgebung

Die Generierung der Karte durch das Testbett wird beim einmaligen Durchfahren des in Abbildung 4.1 gezeigten Parkours umgesetzt. In der Parameterdatei der „SLAM-Toolbox“ kann die Auflösung der Karte in Metern pro Pixel festgehalten werden. Für die erste Karte wird die in der „SLAM-Toolbox“ standardmäßig verwendete Auflösung von 0,05m pro Pixel verwendet.

Die zweite Karte wird mit einer Auflösung von 0,01m pro Pixel generiert. Die hieraus gewonnen Karten lassen sich Abbildung 4.4 und Abbildung 4.3 entnehmen.

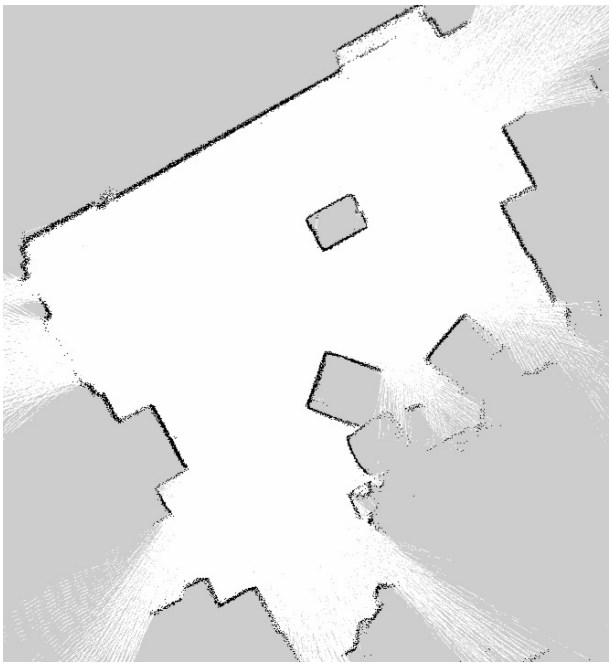


Abbildung 4.3: Generierte Karte bei 0,01m pro Pixel

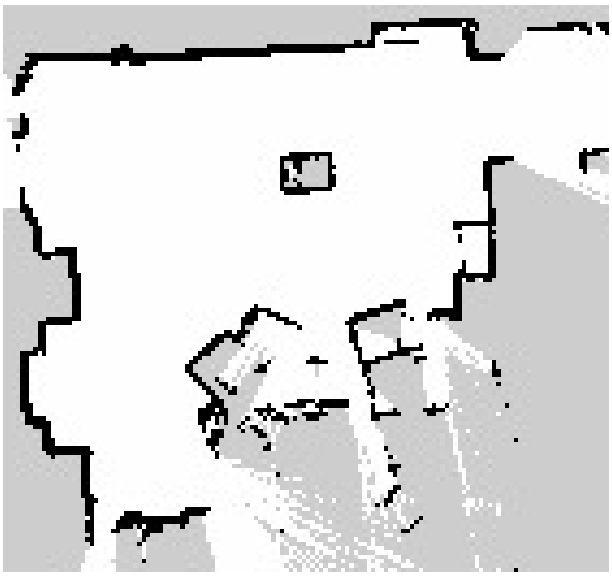


Abbildung 4.4: Generierte Karte bei 0,05m pro Pixel

Die Längen in den dargestellten Karten werden in Microsoft-Paint über die Pixelwerte zwischen den in Abbildung 4.2 dargestellten Längen analysiert. Die in Abbildung 4.2 gezeigten Längen

sind durch Buchstaben gekennzeichnet. Die Messergebnisse der Karten können Tabelle 4.1 entnommen werden.

Tabelle 4.1: Abweichung der Distanzen aus der Vermessung der Karte

Distanz	Länge in Karte 4.3	in Karte 4.4	Wahre Länge
a	5,22m	5,2m	5,21m
b	0,56m	0,55m	0,54m
c	0,38m	0,4m	0,37m
d	1,21m	1,15m	1,26m
e	2,91m	2,85m	2,99m
f	1,43m	1,4m	1,51m

4.1.2 Daten der Positionsbestimmung

Für die Analyse der Positionsbestimmung wurden zwei Ansätze gewählt. Für den ersten Ansatz wird der zeitliche Verlauf der Werte auf der Hauptdiagonalen der Kovarianzmatrix beim Durchfahren der Büroumgebung über die Zeit geplottet. Diese Werte stellen die vom Algorithmus ermittelten Varianzen, der durch die Lokalisierung bestimmten sechs DOF's des Testbetts dar. Da die Ermittlung dieser Varianzen auf dem vom Algorithmus angenommenen Modell basiert, reichen diese alleine nicht als Analysedaten aus. Die Analyse dieser Werte dient hauptsächlich der Unterstützung des zweiten Analyseansatzes. Im zweiten Ansatz der Anlalyse wird eine Referenzposition auf dem Boden der Büroumgebung durch Klebemakierungen festgehalten (siehe Abbildung 4.5).

Abbildung 4.5: Bodenmarkierung der Initialposition beim Start des Versuchs

Nach dem Durchfahren des in Abbildung 4.1 gezeigten Parkours, wird das Testbett zurück an die Referenzposition gefahren und der Positionsversatz in y-Richtung vermessen. Das Chassis des Testbetts liegt beim Start direkt am in Abbildung 4.2 und 4.5 dargestellten Schrank an. Nach dem Abfahren des Parkours wird das Testbett soweit zurückgefahren bis dieses wieder flach am Schrank anliegt. Aus diesem Grund ist nur eine Abweichung der y-Richtung zu genauen Startposition zu vermessen. Hierauf folgend wird die Verschiebung zwischen dem roboterinternen Koordinatensystems (siehe Kapitel 3.4) und des Kartenkoordinatensystems ermittelt. Diese Koordinatentransformationen können aus dem „tf-Topic“ exportiert werden. Dieser Versuch wird insgesamt 5 mal wiederholt und die Verschiebungen zwischen Start- und Endposition, sowie die real gemessene Abweichung ermittelt.

Daten der Kovarianzmatrix über die Zeit

Zur Analyse der Kovarianzmatrix über die Zeit werden die Werte der Hauptdiagonalen über die Zeit dargestellt. Diese stellen die vom Algorithmus bestimmten Abweichungen in der Lokalisierung der DOF's dar. Als Referenz hierfür werden ebenfalls die Daten der Varianzen der Nick- und Rollbewegung geplottet. Diese sollten einen konstanten Wert von Null aufweisen, da diese DOF's nicht von Testbett abgebildet werden. Der zum extrahieren und graphischen Darstellen der Daten verwendete Code ist Anhang B.1 zu entnehmen. Die Plots der Kovarianzmatrix über die Zeit sind Abbildung 4.6 und 4.7 zu entnehmen.

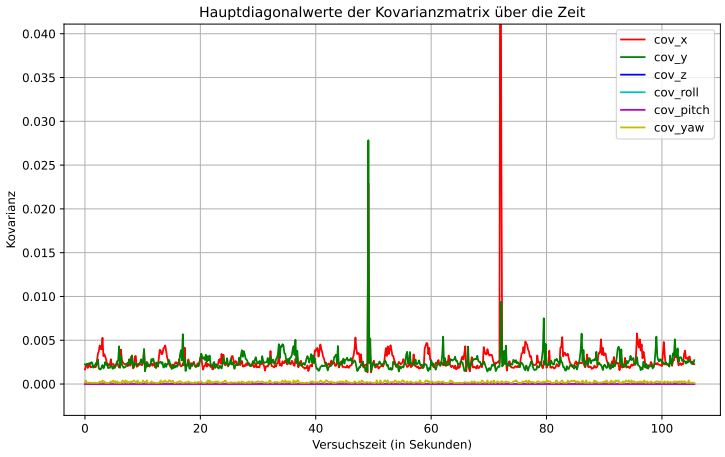


Abbildung 4.6: Kovarianzen über die Zeit bei 0,01m/Pixel

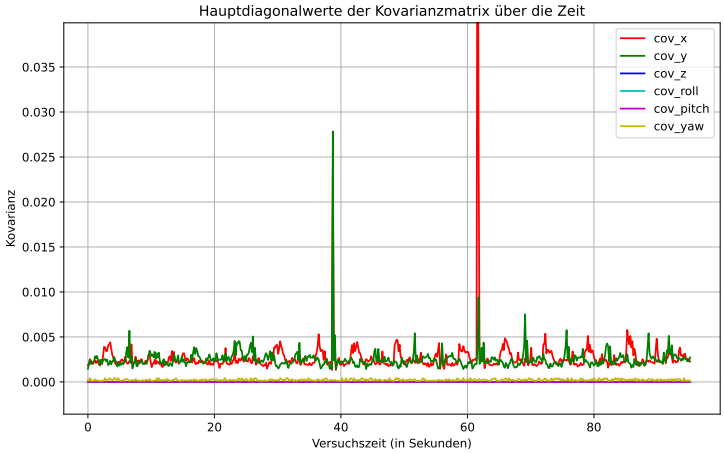


Abbildung 4.7: Kovarianzen über die Zeit bei 0,05m/Pixel

Daten der Verschiebung der Koordinatensysteme

Die durch ROS2 exponierten Koordinatentransformationen beziehen sich auf die Transformation vom Kartenkoordinatensystem auf das Odeometriekoordinatensystem und des Odeometriekoordinatensystems auf das testbettinterne „base_link“-Koordinatensystem. Die Transformationsdaten werden vom „TF-Topic“ exponiert und enthalten den „parent frame“ und „child frame“ (siehe Abbildung 3.9). Das Kartenkoordinatensystem stellt den „parent frame“ des Odeometriekoordinatensystems dar, und das Odeometriekoordinatensystem den „parent frame“ des „base_link“-Koordinatensystems. Ebenfalls enthält diese Nachricht die translatorische Verschiebung der Koordinatensysteme, sowie die Verdrehung als Quaternion ausgedrückt [34].

Zur Bestimmung der Abweichung der Koordinatensysteme zu Beginn und Ende der Messung wird eine Transformationsmatrix verwendet. Die Transformationsmatrix eines Quaternions ergibt sich zu

$$q = (q_0,q_1,q_2,q_3) = \begin{bmatrix} 2(q_0^2+q_1^2)-1 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2) \\ 2(q_1q_2+q_0q_3) & 2(q_0^2+q_2^2)-1 & 2(q_2q_3-q_0q_1 \\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & 2(q_0^2+q_3^2)-1 \end{bmatrix} = \mathbf{T}. \tag{4.1}$$

Durch Multiplizieren des Verschiebungsvektors zwischen dem Odeometrie- und „base_link“-Koordinatensystems mit der Rotationsmatrix kann dieser in das Karten-, Odeometrie-Koordinatensystem überführt werden. Durch Subtraktion dieser beiden Vektoren ergibt sich die Abweichung des Kartenkoordinaten-systems zum „base_link“-Koordinatensystem. Der zur Berechnung verwendete Python-Code kann Anhang B.1 entnommen werden. Die Abweichungen aus

den fünf Versuchen bei unterschiedlicher Kartenauflösung sind Tabelle 4.2 und 4.3 zu entnehmen. Die Mittelwerte für die Verschiebung in y-Richtung wurde mit den bereinigten y-Werten angestellt.

Tabelle 4.2: Abweichung der Positionsdaten für 0,01m/Pixel und Mittelwerte (Versuch 5 ist exkludiert)

Versuch	Abweichung x in Meter	Abweichung y in Meter	Verschiebung in y-Richtung
1	-0,0074m	-0,0435m	-0,036m
2	0,006m	-0,012m	-0,006m
3	0,007m	-0,039m	-0,027m
4	0,001m	-0,026m	-0,018m
5	-1,931m	1,002m	0,026m
Mittelwerte	0,005m	0,011m	

Tabelle 4.3: Abweichung der Positionsdaten für 0,05m/Pixel und Mittelwerte

Versuch	Abweichung x in Meter	Abweichung y in Meter	Verschiebung in y-Richtung
1	-0,038m	-0,0368m	-0,006m
2	0.008m	-0.040	-0,028m
3	0,001m	-0,004	0m
4	0,009m	-0,045m	-0,031m
5	0,003m	-0,011m	-0,005m
Mittelwerte	0,0118m	0,0133m	

4.1.3 Auswertung des simultanen Lokalisierens und Kartographierens zu Positionsbestimmung

In diesem Kapitel folgt eine Gegenüberstellung und Auswertung der in Kapitel 4.1.1 und 4.1.2 generierten Daten.

In Anbetracht der Daten aus Tabelle 4.1 im Kapitel 4.1.1 fällt auf, dass die Abbildung der wahren Länge der Raumgrößen genauer mit der Kartenauflösung von 0,01 Meter pro Pixel messbar sind. Auffällig ist jedoch, dass die Raumgrößenmessungen nicht in jeder Messung die erwarteten Abweichungen von $\pm$0,05m und $\pm$0,01m aufweisen. Die Raumgrößen der Messung mit der Auflösung von 0,01 Metern pro Pixel sind zwar genauer, jedoch nicht signifikant genauer. Grund hierfür könnte die Geschwindigkeit sein in welcher die Kartendaten generiert werden.

Auffällig bei der Durchführung des Versuches war, dass die Geschwindigkeit, in welcher die Karte generiert wurde, stark von der Auflösung der Karte abhängig ist. Hierbei gilt je höher die Auflösung, desto langsamer die Generierung der Karte. Dies lässt sich ebenfalls den Abbildungen 4.3 und 4.4 entnehmen. Auch wenn die Darstellung von Geraden innerhalb der Büroumgebung in Abbildung 4.3 genauer erscheint, sind die Geraden (beispielsweise der Wände) nicht vollständig gerendert und weisen Lücken auf. Dieser Effekt wäre beispielsweise durch höhere Rechenleistung der Hardware zur zentralen Datenverarbeitung (siehe Kapitel 3.2.3) mitigierbar. Die Geschwindigkeit der Erstellung der Karte in Abhängigkeit von der Auflösung steht in direkter Korrelati-

on zur Leistungsfähigkeit des GPUs und CPU's. Ein weiterer Aspekt zwischen den Karten aus Abbildung 4.3 und 4.4 ist die Bildung von Artefakten. Artefakte stellen in digitalen Bildern Anzeigefehler dar, welche nicht ihren Ursprung in den Ausgangsdaten haben. Diese Artefakte können beispielsweise in Abbildung 4.4 an dem implementierten Hindernis gesehen werden. Die Karte stellt hierbei ein Objekt an der Begrenzung des Hindernisses dar, obwohl sich dort keines befindet. Des Weiteren sind Artefaktbildungen an den Wänden der Büroumgebung in Abbildung 4.4 sichtbar. Die Menge und Größe dieser Artefakte ist hierbei, wie in Abbildung 4.3 im Vergleich zu Abbildung 4.4 zu sehen, abhängig von der Auflösung der Karte.

Bei Betrachtung der Plots der Kovarianzen aus Abbildung 4.6 und 4.7 wird deutlich, dass die vom Algorithmus bestimmten Unsicherheiten in der Bestimmung der Gierung, x- und y-Koordinaten unabhängig von der Auflösung der Karte sind. Diese belaufen sich im Schnitt innerhalb des Bereiches 0,002 und 0,006. Die Kovarianzwerte können hierbei Werte zwischen 0 und 1 annehmen und tragen keine Einheit. Die Kovarianzen der Roll- und Nickbewegung, welche als Referenz geplottet wurden, befinden konstant beim Wert 0. Diese Beobachtungen sprechen nicht für eine Korrelation der Auflösung mit der estimierten Unsicherheit in der Positionsbestimmung. Die Orte der Peaks der Unsicherheiten der Gierung, x- und y-Koordinate sprechen für ein systematisches Phänomen, da diese über die beiden Versuche im gleichen Abstand zueinander auftreten. Grund hierfür könnte der Ort der Kartengenerierung zum Zeitpunkt des Auftretens der Peaks sein. An dem Zeitpunkt des Auftretens des x-Koordinatenpeaks findet das Mappen des Flurs (siehe Abbildung 4.2) statt, wobei die Länge des Flurs länger als die Maximalreichweite des LIDAR-Sensors ist. Somit enthält zu diesen Zeitpunkt die Messung des LIDAR-Sensors eine höhere Menge an nicht ermittelten Abstandsdaten, was zu einer temporären höheren Unsicherheit führen könnte. Die Signifikanz dieser Peaks in der Positionsbestimmung stellt sich jedoch als gering dar, da die Peaks innerhalb der Kovarianzen Matrix einen Maximalzeitraum von jeweils 0,2 Sekunden abdecken.

Der Bestimmung der Positionsabweichung zwischen Start- und Endposition (siehe Tabelle 4.2 und 4.3) ist zu entnehmen, dass trotz der auftretenden Peaks die Positionsbestimmung zur Referenzposition im Bereich zwischen 0,005m bis 0,0133m genau ist. Zur Berechnung der Mittelwerte der y-Abweichung in Tabelle 4.2 wird der fünfte Messwert exkludiert, da dieser aufgrund seiner sehr hohen Abweichung auf einen temporären Messfehler hindeutet. Das Vorhandensein dieses Messfehlers innerhalb der Messreihe deutet jedoch auf Probleme in der Deterministik des Algorithmus hin, da diese sehr große Abweichung in den Transformationsdaten zu großen Positionsabweichungen innerhalb der Karte führt. Die in Tabelle 4.2 und 4.3 gezeigten Mittelwerte sprechen für eine Erhöhung der Auflösung der Position in x-Richtung bei Erhöhung der Auflösung der Karte. Die kaum vorhandene Erhöhung der Auflösung der Position in y-Richtung könnte durch einen systematischen Fehler innerhalb des Lokalisierungsalgorithmus sprechen, oder für Messabweichungen in der Ermittlung der Enddifferenz zur Referenzposition. Um die Akkuratheit der Lokalisierung genauer zu untersuchen, könnten größere Messreihen angelegt sowie ein genaueres System zur Bestimmung der Differenz zwischen Start- und Endposition implementiert werden. Hierfür könnte zusätzliche Sensorik wie in Kapitel 2.2 verwendet wer-

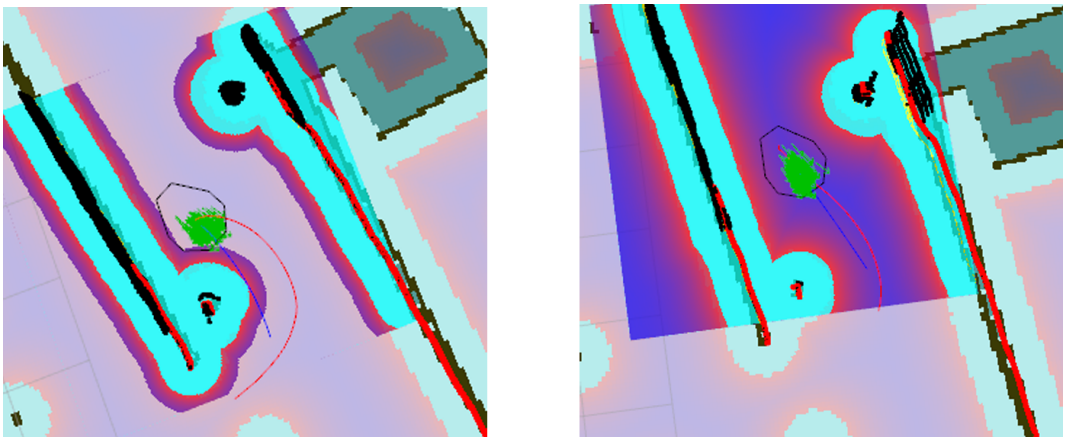
den, um die Auflösung der Positionsdaten weiter zu verbessern, sowie die genaue Endposition besser zu erfassen. Hierfür könnten PDOA-Methoden mit UWB (siehe Kapitel 2.1.3) oder „Motion Tracking“-Kamerasysteme im Kontext von Auto-Ident-Systemen (siehe Kapitel 2.2.3 und 2.2.9) benutzt werden. Zusammengefasst sprechen die Daten für eine ausreichend genaue Abbildbarkeit der Lokalisierungsfähigkeiten zur Umsetzung von Nahbereichsmanövern (Kapitel 3.1 Grundvoraussetzung 2).

4.2 Autonome Navigation zur Annäherung im Nah und Fernbereich

Zur Untersuchung der autonomen Navigation des ROS2 internen „NAV2“-Pakets wird ein qualitatives Modell verwendet. Als Grundlage dieses Modells dient der in Abbildung 4.1 definierte Parkour. „NAV2“ besitzt eine große Anzahl an Parametern und verschiedenen „Plugins“, welche zum Optimieren des Navigationsverhaltens genutzt werden können. Aufgrund des Umfangs des „NAV2“-Pakets werden zur Optimierung lediglich die Parameter „inflation_radius“, „cost_scaling_factor“ der globalen und lokalen „costmap“ verwendet. Die globale „costmap“ wird von „NAV2“ verwendet, um innerhalb der durch die „SLAM-Toolbox“ generierte Karte eine Trajektorie zu einem Ziel zu planen [26], und umfasst den gesamten Kartenbereich. Die lokale „costmap“ beschränkt sich auf einen fest definierten Bereich um das Testbett herum und weist eine höhere Updaterate auf [26]. Diese wird verwendet, um zwischen lokalen, statischen und dynamischen Hindernissen zu navigieren [26]. Die Differenzierung in eine lokale und globale „costmap“ hat den Vorteil, dass nur für die unmittelbare Umgebung dynamische Berechnungen angestellt werden müssen, was den Rechenaufwand des „NAV2“-Pakets verringert [26].

Der „inflation_radius“ beschreibt hierbei den Radius der „costmap“, welche um detektierte statische und dynamische Hindernisse generiert wird. Der „cost_scaling_factor“ beschreibt einen exponentiellen Verfallsfaktor welcher vorgibt wie sich die zugewiesenen Kosten über den „inflation_radius“ verhalten. Hierbei gilt je größer der „cost_scaling_factor“, desto schneller sinken die Kosten über den „inflation_radius“. Diese Werte müssen anhand der Größenordnungen des Raumes angepasst werden. Ziel der Anpassung ist es, dass die „inflatio_radii“ im schmalsten Bereich des Parkours nah aneinander liegen. Der „cost_scaling_factor“ soll hierbei eine moderate Abnahme der Kosten über den Radius erreichen [55, S.12]. Grund hierfür ist, dass die Routenplanung des Navigationsalgorithmus möglichst eine Route wählt, die weit von Begrenzungen links und rechts des Testbetts verläuft (siehe Abbildung 4.8). Diese Routen werden im Optimalfall um die Ränder der „costmap“ der zu navigierenden Objekte gelegt.

Die richtige Auswahl dieser Werte ist stark durch Erfahrungswerte, sowie die Raumgrößen geprägt. Werte die sich für die Büroumgebung als passend dargestellt haben sind Tabelle 4.4 zu entnehmen.



(a) Steile Inflationskurve: mit naher Trajektorie an den Begrenzungen

(b) moderate Inflationskurve: mit größerer Entfernung zu den Begrenzungen

Abbildung 4.8: Abbildung verschiedener Trajektorien bei varierenden „cost_scaling_factors“: Die rote Linie stellt die geplante und die blaue Linie die tatsächliche Trajektorie dar [55, S.12]

Tabelle 4.4: NAV2 Parameterdaten für die Büroumgebung

costmap Typ	inflation_radius	cost_scaling_factor
global	0,53m	5,5
local	0,53m	5,3

4.2.1 Daten der autonomen Navigation

Zur Analyse der autonomen Navigationsfähigkeiten durchfährt das Testbett fünf mal den Parkour bei unterschiedlichen Kartenauflösungen. Als qualitative Metriken werden die Fähigkeit betrachtet den Parkour durchfahren zu können, wie häufig das Testbett während des Durchfahrens in ein „Recovery“-Verhalten wechselt (siehe Kapitel 3.4), sowie die Zeit, die benötigt wurde, um den Parkour zu absolvieren. Zur Umsetzung der Navigationsaufgabe wird das Feature „navigation through waypoints“ verwendet. Dieses Feature ermöglicht es feste Wegpunkte innerhalb der Karte zu setzten, welche durchfahren werden müssen. Die gesetzten „waypoints“ stellen vektorielle Größen dar und werden für jeden Versuch an die selben Stellen in der Karte gesetzt. Vor dem Durchfahren des Parkours wird dieser ein Mal manuell durchfahren um die für die Navigation nötige Karte zu generieren, hierauf folgend startet der Versuch.

Die Ergebnisse dieser Versuchsreihe sind Tabelle 4.5 und 4.6 zu entnehmen.

Die Mittelwerte der Spalte „Parkour absolviert“ ergeben sich, anhand der gefahrenen Parkours, zu der Wahrscheinlichkeit, dass der Parkour absolviert wird. Bei der Berechnung des Mittelwerts wird ein Absolvieren des Parkours als 1 bewertet und ein Nichtabsolvieren als 0.

Tabelle 4.5: Daten der Navigationsversuche bei 0,05m pro Pixel

Versuch	Parkour absolviert	Anzahl der Recoverys	Zeit
1	Nein	15	4:17min
2	Ja	7	2:41min
3	Ja	7	2:30min
4	Ja	8	2:36min
5	Ja	5	2:33min
Mittelwerte	0,8	8,4	2:55min

Tabelle 4.6: Daten der Navigationsversuche bei 0,01m pro Pixel

Versuch	Parkour absolviert	Anzahl der Recoverys	Zeit
1	Ja	3	2:21min
2	Ja	3	2:20min
3	Ja	5	2:34min
4	Ja	2	2:10min
5	Ja	4	2:19min
Mittelwerte	1	3,4	2:21min

4.2.2 Auswertung der autonomen Navigation

In Anbetracht der in Tabelle 4.5 und 4.6 generierten Werte wird eine Korrelation zwischen der Anzahl der „Recoverys“ und der Auflösung der für den Versuch zugrunde liegenden Karte deutlich. Bei Erhöhung der Auflösung der Karte um den Faktor 5 sinkt die durchschnittliche Anzahl der „Recoverys“ um den Faktor 2,47. Grund hierfür könnte die Reduktion der in Kapitel 4.1.3 beschriebenen Artefakte und die daraus resultierende Vergrößerung der freien Flächen innerhalb der Karte sein. Die verbesserte Auflösung der Karte und der daraus resultierenden verbesserten Auflösung der „costmap“ ermöglicht hierbei eine weniger riskante Routenplanung des Testbetts.

In Anbetracht der durchschnittlichen Zeiten des Absolvierens des Parkours (siehe Tabelle 4.5 und 4.6) ist eine Reduktion der Durchschnittszeit um 34 Sekunden bemerkbar. Grund hierfür ist die verringerte Anzahl an „Recoverys“. Beim Auslösen des „Recovery“-verhaltens (beschrieben in Kapitel 3.4) führt das Testbett eine Drehung durch, was die Gesamtdauer zur Absolvierung des Parkours deutlich erhöht. Des Weiteren kann das Durchführen dieser Drehung innerhalb enger Passagen des Parkours zu einer Manövrierunfähigkeit des Testbetts führen. Dieser Fall ist in Tabelle 4.5 beim ersten Versuch abgebildet. Eine Verbesserung der Zeit welche zum Absolvieren des Parkours notwendig ist, sowie eine Reduktion der Anzahl der „Recoverys“ könnte voraussichtlich durch eine ganzheitliche Anpassung der Navigationsparameter erfolgen. Grund für diese Annahme ist, dass der verwendete Algorithmus bereits in verschiedenen industriellen Bereichen kommerzielle Anwendung findet [25]. Um die Deterministik des Algorithmus näher zu untersuchen wären größere Versuchsreihen, sowie eine größere Anzahl an verschiedenen Parkours notwendig. Die Untersuchung der Deterministik ist jedoch nicht Ziel dieser Abschlussarbeit.

4.3 Fazit der Auswertung

Die in Kapitel 4.1.1 und 4.2.1 durchgeführten Versuchsreihen zeigen die Nutzbarkeit der verwendeten Algorithmen zur Lokalisierung und Navigation innerhalb der in Kapitel 3.1 unter Randbedingung 3 definierten Büroumgebung.

Die im Kapitel 4.1.1 gezeigten Genauigkeiten der Positionsbestimmung zeigten die für ein autonomes Docking ausreichende Auflösung von ± 0,01m (siehe Kapitel 3.1 Gundvorraussetzung 2). Die vom Algorithmus bestimmten Varianzen der Positionsbestimmung über die Zeit zeigen keine direkte Abhängigkeit zur Auflösung der Karte. Wie genau sich die Positionsbestimmung im Verhältnis zur wahren Position über die Zeit verhält, bleibt für einen Abgleich der vom Algorithmus estimierten Varianzen offen. Zur genaueren Bestimmung der wahren Position könnten Methoden wie eine externe kalibrierte Stereo-Kamera (siehe Kapitel 2.2.3) umgesetzt werden, um eine genaue Referenz zur wahren Position über die Zeit zu erhalten. Zusätzlich könnten Auto-Ident-Systeme (siehe Kapitel 2.2.9) genutzt werden um mit Markierungen auf dem Testbett, oder RFID-Systemen die genaue Ausrichtung und Entfernung zu einer festgelegten Endposition zu bestimmen. Die Genauigkeit der Kartographierungsfähigkeiten sind neben der Auflösung der generierten Karte von der Genauigkeit des LDIAR-Sensors abhängig. Ebenfalls ist diese von der Genauigkeit der zeitlichen Auflösung der vom LIDAR-Sensor übermittelten Daten abhängig, welche in dieser Abschlussarbeit nicht näher untersucht wurde.

Die in Kapitel 4.2.1 gezeigten Daten der autonome Navigationsfähigkeit des Testbetts ist ausreichend um einen komplexen Parkour zu absolvieren. Die im Kapitel 4.2.1 ermittelten Daten sprechen für eine Korrelation der Auflösung der Karte mit der Navigationsfähigkeit sowie der für den Parkour notwendigen Zeit. Die hierfür nötige Zeit sowie die Deterministik mit welche der Parkour zu absolvieren ist, könnte weiter mit einer genaueren Ermittlung der Navigationsparameter verbessert werden. Eine genauere Untersuchung Docking spezifischer Betriebsmodi im autonomen Navigieren bleibt hierbei offen, da die genaue Abbildung dieser nicht Teil der Abschlussarbeit ist.

Auf Grundlage der Güte der Daten der Positionsbestimmung, sowie der Fähigkeit der autonomen Navigation lässt sich eine gegenseitige Lokalisierung zweier Systeme, welche sich innerhalb der gleichen Karte befinden, umsetzten. Die hierfür notwendigen Positionsdaten können hierbei untereinander über ROS2-Dienste ausgetauscht werden, um die aktuelle Position des jeweils anderen Systems als Ziel zu verwenden. Eine weitere mögliche Umsetzung wäre hierbei ein stationäres Ziel mit fester Position innerhalb der Karte. Ein dynamisches System würde sich dem stationären Ziel autonom mit einer vorher definierten Endausrichtung und -position annähern. Beide dieser Szenarien sind im Kontext von „Systems of Systems“-Anwendungen denkbare Anwendungsfälle.

5 Zusammenfassung und Ausblick

Das Ziel der vorliegenden Arbeit war die Integration einer sensorgestützten Lösung innerhalb eines Testbetts zur gegenseitigen Lokalisierung und autonomen Annäherung zweier Objekte für künftige Weltraumanwendungen umzusetzen.

In Kapitel 2.1 dieser Abschlussarbeit wurden die physikalischen Grundlagen verschiedener Sensortechnologien dargelegt. Ziel dieses Kapitels war es, die Grundlagen, auf welchen verschiedene Sensortechnologien für Lokalisierungsprobleme operieren, zu beschreiben. Grund hierfür war einen Überblick über die möglichen physikalischen Lösungsansätze zu schaffen.

Kapitel 2.2 beschäftigte sich hierauf folgend mit verschiedenen in der Raumfahrt eingesetzten Sensoren, welche die in Kapitel 2.1 dargestellten physikalischen Grundlagen zur Lokalisierung und Abstandsbestimmung nutzen. Dieses Kapitel stellte die Grundlage einer Auflistung und Auswahl an Sesorik, welche zur Implementierung der Lokalisierungs- und Navigationsfähigkeiten des Testbetts in Frage kommt.

Kapitel 3 diente der Darlegung der Anforderungen und Grundvoraussetzung des Testbetts. Diese Darlegung ist notwendig, um den Rahmen einer Abbildbarkeit des Testbetts für zukünftige Weltraumanwendungen zu schaffen, sowie die Vereinfachungen festzuhalten, welche im Zuge dieses Rahmens getroffen wurden. Im weiteren Verlauf des Kapitels wurden verschiedene mögliche Hardwarekomponenten gegenübergestellt, um eine Auswahl der für im Testbett zu implementierenden Komponenten zu treffen. Hierbei wurde in Hardware unterschieden, welche als Vorgabe vom DLR gestellt wurde und Hardware, welche frei wählbar war. Um eine differenzierte Hardwareauswahl zu treffen wurden verschiedene Systeme einander gegenübergestellt, sodass eine fundierte Auswahl der Komponenten getroffen werden kann.

Das Kapitel 3.4 befasst sich mit der im Testbett verwendeten Software. Diese soll in der Lage sein, die in Kapitel 3 ausgewählten Hardwarekomponenten mit einander zu verbinden. Ebenfalls war es Aufgabe in dieser Software die im Titel dieser Abschlussarbeit festgehaltenen Lokalisierungs- und autonomen Navigationsfähigkeiten zu implementieren.

Abschließend befasste sich Kapitel 4 mit dem Test der Lokalisierungs- und autonomen Navigationsfähigkeit des Testbetts. Hierfür wurden verschiedene statistische Methoden herangezogen, um diese Fähigkeiten quantitativ und qualitativ zu analysieren. Ergebnis dieser Analyse war, dass die implementierten Lokalisierungsfähigkeiten ausreichend sind, um zwei Systeme innerhalb einer generierten Karte zu lokalisieren. Diese Positionsdaten können über ROS2-„Topics“

kommuniziert werden, um die aktuellen Positionsdaten der Systeme mit einander zu teilen. Die Positionsdaten wiesen eine maximale mittlere Auflösung von $\approx$ 0,01m auf. Hieraus folgt, dass nach der Bedingung in Kapitel 3.1 unter Grundvoraussetzung 2 gestellten Auflösung diese ausreichend ist, um Nahbereichsmanöver zu ermöglichen. Als Weiterführung dieser Arbeit wäre eine Implementierung mehrerer Sensoren, sowie die Sensordatenfusion zur Verbesserung der Auflösung der Lokalisierung von Interesse. Zu dem könnte eine leistungsfähigere Hardware zur zentralen Datenverarbeitung die Geschwindigkeit in welcher die Karten generiert werden deutlich verbessern. Eine nähere Analyse der Navigationsparameter könnte die Navigationsfähikeit mittels des verwendeten ROS2-Pakets ebenfalls verbessern. In welchem Umfang die in Kapitel 2.2 genannten Sensoren ,welche nicht für einen Einsatz innerhalb einer Büroumgebung geeignet sind, eine bessere Auflösung der Lokalisierung für Weltraumanwendungen bieten, bleibt Thema zukünftiger Forschung.

Um eine genaue Quantifizierung der Abbildbarkeit des Testbetts zu beispielsweise Satelliten zu stellen, wäre es notwendig die Lokalisierung und autonome Annäherungen im dreidimensionalen Raum zu untersuchen. Die Reduktion des Testbetts auf zwei DOF's ist eine für terrestrische Tests notwendige Vereinfachung, diese weist jedoch das Problem auf, dass sie die restlichen vier DOF'S nicht mit abbildet. In welchem Ausmaß die untersuchte Algorithmik mit sechs DOF's anwendbar ist, bleibt eine zukünftige Forschungsfrage. Durch eine Reduktion der Randbedingungen aus Kapitel 3.1 eröffnet sich zusätzlich die Möglichkeit Sensorik zu testen welche in dieser Abschlussarbeit auf Grund der Randbedingungen exkludiert wurde. Diese Abschlussarbeit dient als „Proof of Concept“ in welchem Ausmaß die verwendete Algorithmik und Sensorik in der Lage ist die in den Versuchen untersuchten zwei DOF's abzubilden.

Eine zukünftige Weiterentwicklung des ROS2-„Frameworks“ für Raumfahrtanwendungen bleibt eine fortwährende Aufgabe. Die von der NASA bestrebten Verbesserungen im Zuge der Entwicklung von „SPACE-ROS“ beschäftigen sich aktuell mit einigen der in dieser Abschlussarbeit aufgezeigten Probleme. Hierzu zählen die Deterministik der ROS2-Pakete, sowie Standardisierungen von Hardwaretreibern, welche häufig zu Limitationen innerhalb der Genauigkeit von verschiedenen Abläufen führen. Im Allgemeinen bietet ROS2 ein sehr umfängliches „Open Source-Framework“ zur Entwicklung verschiedenster Robotikanwendungen. Dies stellt auf Grund des „Open Source“-Aspekts ein großes Potenzial für zukünftige Entwicklungen im Bereich der Robotik sowie im Kontext von Raumfahrtanwendungen dar.

Literatur

[1] Adamy, D.: a second course in electronic warfare, artech house, 2004.

[2] Anet, C.: Why is Dynamic Range so inmportant, https://blogs.qsc.com/live-sound/why-is-dynamic-range-so-important/ (aufgerufen 10. Januar 2025), 2023.

[3] Arduino: Arduino Nano, https://docs.arduino.cc/resources/datasheets/A000005-datasheet.pdf (aufgerufen 2. Dezember 2024), 2024.

[4] Bidikar, B.; Rao, G.; Laveti, G.; Kumar, M.: Satellite Clock Error and Orbital Solution Error Estimation for Precise Navigation Applications. Positioning, 5, 22-26. 2014.

[5] Boquet, G.; Boquet-Pujadas, A.; Pisa, I.; Dabak, A.; Vilajosana, X.; Martinez, B.: Indoor position estimation using angle of arrival measurements: An efficient multi-anchor approach with outlier rejection. Internet of Things 26 (2024) 101236, 2024.

[6] Curtis, H.: Digital Slit Sun Sensor, https://blog.satsearch.co/2020-02-12-sun-sensors-an-overview-of-systems-available-on-the-global-marketplace-for-space (aufgerufen 10. Januar 2025), 2019.

[7] Duch, R.: Phases and Manoeuvres Involved in Orbital Rendezvous Missions. International Journal for Research and Development in Technology, 2016.

[8] Gross, D.; Hauger, W.; Schröder, J.; Wall, W.: Technische Mechanik 1, Springer, 2013.

[9] Haun, M.: Handbuch Robotik, Springer Vieweg, 2013.

[10] hb Robotics: Arduino H-Bridge Driver, https://github.com/hbrobotics/ros_arduino_bridge (aufgerufen 14. Januar 2025), 2023.

[11] Hering, E.; Schönfelder, G.: Sensoren in Wissenschaft und Technik, Springer, 2018.

[12] Hinkler, H.; Strube, M.; Zipay, J.; Cryan, S.: Technology Development of Automated Rendezvous and Docking/Capture Sensors and Docking Mechanism for the Asteroid Redirect Crewed Mission. IEEE Aerospace Conference, 2016.

[13] Höver, N.; Lichte, B.: Multi-beam Lidar Sensor for Active Safety Applications. SAE Paper 06AE-138, Transactions Journal of Passenger Cars, 2006.

[14] Howard, R.; Bryan, T.; Brewster, L.: Proximity Operations and Docking Sensor Development. IEEE Aerospace conference, 2009.

[15] INNO-Maker: Super Mini TDOF Principle Lidar, https://gibbard.me/lidar/datasheet.pdf (aufgerufen 29. Dezember 2024), 2024.

[16] Institute of Atmospheric Physics: Digital Slit Sun Sensor, https://www.ufa.cas.cz/en/institute-structure/department-of-ionosphere-and-aeronomy/digital-slit-sun-sensor/ (aufgerufen 3. Januar 2025), 2019.

[17] ITEM Industrietechnik: ITEM Konstruktionsprofile, https://www.item24.com/de-de/profiltechnik (aufgerufen 10. Januar 2025), 2025.

[18] Jena-Optronik: Space-Qualified Scanning LIDAR for Rendezvous and Docking Applications, 2022.

[19] Jena-Optronik: The RVS3000 and RVS3000-3D LIDAR Sensors, 2016.

[20] Jena-Optronik: The RVS3000 and RVS3000-3D LIDAR Sensors: Recent Technological Advances and Future Applications, 2017.

[21] LDRobots: LD06 ROS2 Package, https://github.com/ldrobotSensorTeam/ldlidar_ros2 (aufgerufen 2. Januar 2025), 2022.

[22] LUX-onis: OAK-D Lite, https://shop.luxonis.com/products/oak-d-lite-1 (aufgerufen 2. Januar 2025), 2022.

[23] Macenski, S.; Foote, T.; Gerkey, B.; Lalancette, C.; Woodall, W.: Robot Operating System 2: Design, architecture, and uses in the wild. Science Robotics, Bd. 7, Nr. 66, eabm6074, 2022.

[24] Macenski, S.; Jambrecic, I.: SLAM Toolbox, https://github.com/SteveMacenski/slam_toolbox?tab=readme-ov-file (aufgerufen 5. Januar 2025), 2021.

[25] Macenski, S.; Jambrecic, I.: SLAM Toolbox: SLAM for the dynamic world. Journal of Open Source Software, Bd. 6, Nr. 61, S. 2783, 2021.

[26] Macenski, S.; Martín, F.; White, R.; Ginés Clavero, J.: The Marathon 2: A Navigation System, https://arxiv.org/pdf/2003.00368 (aufgerufen 10. Januar 2025), 2020.

[27] Mongrard, O.; Ankers, F.; Casiez, P.; Cavrois, B.; Donnard, A.; Vergnol, A.; Southivong, U.: LIRIS flight database and its use toward non cooperative rendezvous. Progress in Flight Dynamics, Guidance, Navigation, and Control, 2018.

[28] Newans, J.: my_bot, https://github.com/joshnewans/my_bot (aufgerufen 10. Januar 2025), 2020.

[29] Newans, J.: ROS Arduino Bridge, https://github.com/joshnewans/diffdrive_arduino (aufgerufen 14. Januar 2025), 2023.

[30] Nolet, S.: Development of a Guidance, Navigation and Control Architecture and Validation Process Enabling Autonomous Docking to a Tumbling Satellite, 2007.

[31] Northrop Grumman: LN-200 Fiber Optic Inertial Measurement Unit, www.nast-group.caltech.edu/~murray/dgc05/upload/7/7f/Inertial_Measurement_Unit_LN-200.pdf (aufgerufen 3. November 2024), 2016.

[32] Nvidia: Nvidia Jetson AGX Origin, www.nvidia.com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf (aufgerufen 2. Dezember 2024), 2024.

[33] Nvidia: Nvidia Jetson Nano, https://openzeka.com/en/wp-content/uploads/2022/08/JetsonNano_DataSheet_DS09366001v1.1-1.pdf (aufgerufen 2. Dezember 2024), 2024.

[34] Open Robotics: ROS Wiki, https://wiki.ros.org/ (aufgerufen 2. Januar 2025), 2022.

[35] Pagels-Kerb, A.: Lecture notes in Spacecraft Assembly 2, 2023.

[36] Parallax Inc.: Motor Mount and Wheel kit Datasheet, 2015.

[37] Poulose, A.; Kim, J.; Han, S.: A Sensor Fusion Framework for Indoor Localization Using Smartphone Sensors and Wi-Fi RSSI Measurements. Applied Science 2019, 9, 4379, 2019.

[38] Princeton University: Global Navigation Satellite System, www.princeton.edu/~alaink/Orf467F07/GNSS.pdf (aufgerufen 3. November 2024), 2007.

[39] Rao, R.: Computer Vision, 2009.

[40] Raspberry Pi Foundation: Raspberry Pi 4 B, https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/ (aufgerufen 2. Dezember 2024), 2024.

[41] Raspberry Pi Foundation: Raspberry Pi 5, https://www.raspberrypi.com/products/raspberry-pi-5/ (aufgerufen 2. Dezember 2024), 2024.

[42] Raspberry Pi Foundation.: Raspberry Pi Pico Datasheet, https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf (aufgerufen 12. Dezember 2024), 2024.

[43] Rauf, J.: SpaceX Dragon Spacecraft, 2023.

[44] Ruel, S.; Luu, T.; Berube, A.: Space Shuttle Testing of the TriDAR 3D Rendezvous and Docking Sensor. journalof field Robotics–2012, 2011.

[45] Skolnik, M.: RADAR HANDBOOK, MC Graw Hill, 2008.

[46] StereoLabs: ZED 2 Camera and SDK Overview, https://www.generationrobots.com/media/zed2-camera-datasheet.pdf (aufgerufen 29. Dezember 2024), 2024.

[47] Takugo, A.: The Design and Operational Principle of Star Trackers, https://bpb-us-e1.wpmucdn.com/sites.psu.edu/dist/4/32637/files/2018/04/Takugo-AmyThe-Design-and-Operational-Principle-of-Star-Trackers-Finaldraft-1k5e4lp.pdf (aufgerufen 3. November 2024), 2018.

[48] Teensy: Teensy 40, https://www.pjrc.com/store/teensy40.html (aufgerufen 10. Dezember 2024), 2024.

[49] U.S. Department of Defence: Global Positioning System Standard, Positioning Service Performance Standard, 2020.

[50] Winter, H.; Hakuli, S.; Lotz, F.; Singer, C.: Handbuch Fahrerassistenzsysteme, Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort, Springer Vieweg, 2015.

[51] Woodall, W.; Harrison, J.: Serial Communication Library, https://github.com/cottsay/serial (aufgerufen 2. Januar 2025), 2018.

[52] Zentrum für angewandte Raumfahrttechnologie und Microgravitation: Fluxgate Magnetometer, https://www.zarm-technik.de/ (aufgerufen 3. November 2024), 2020.

[53] Zhang, Y.; Duan, L.: A phase-difference-of-arrival assisted ultra-wideband positioning method for elderly care. Measurement Volume 170, January 2021, 108689, 2021.

[54] Zhao, W.; Cheng, Y.; Zhao, S.; Hu, X.; Rong, Y.; Duan, J.; Chen, J.: Navigation Grade MEMS IMU for A Satellite. micromachines 2021, 12, 151, 2021.

[55] Zheng, K.: ROS Navigation Tuning Guide, 2016.

A Technologien

A.1 Anwendungen zur Lokalisierung in Nah und Fernbereich

Ziel dieses Teils des Anhangs ist es einige Beispiele zur Anwendung von Sensortechnologien zur Lokalisierung im Nah- und Fernbereich zu geben. Hierbei geht es vor allem um Sensorsysteme welche bereits erfolgreich auf Flugmissionen getestet und validiert wurden.

A.1.1 RVS3000(-3D)

Bei dem RVS3000 und RVS3000-3D handelt es sich um operative Sensoren des Unternehmens Jena-Optronik, welche bereits in Systemen wie dem „ATV“ der ESA, dem „Dream Chaser“ und dem japanischen „HTV“ [19] als verbaute Sensorlösungen zum Einsatz kommen. Zu den von Jena-Optronik definierten zukünftigen und jetzigen Einsatzbereichen für 3D-LIDAR Sensorik zählen „On-Orbit Servicing“, „Debris Removal“, „In-Orbit Assembly“ und „Planatary Landing“ [20]. Das RVS3000 und RVS3000-3D sind Sensorsysteme, welche mittels LIDAR-Sensorik 3D-Punktwolkendaten generieren um aus diesen Informationen über die Umgebung und die in ihr befindlichen Objekte, sowie ihren Abstand zum Sensor zu generieren.

Das RVS3000 und RVS3000-3D unterscheiden sich einerseits in der Operating Range, sowie in der Intention zur Anwendung. Das RVS3000 ist für kooperative Ziele ausgelegt, somit für Ziele welche über Reflektoren oder andere Sensorik verfügen um mit dem RSV3000 Interagieren zu interagieren [20]. Der Einsatzbereich des RVS3000 beläuft sich auf 1-3000m während das RVS3000-3D eine Operating Range von 1-1500m aufweist [20]. Einer der Hauptunterschiede besteht außerdem in den internen Verarbeitungseinheiten. Die internen Verarbeitungseinheiten des RVS3000-3D sind FPGA basiert um den erhöhten Leistungsbedarf zum Image Processing zu ermöglichen. Während das RVS3000 nur Datenverarbeitung für die Reflektoren des kooperativen Ziels ermöglichen muss [20]. Das RVS3000-3D stellt somit eine „One Box Solution“ [20] dar, welches Bildverarbeitung und LIDAR-Sensorik mit einander verbindet. Dies ermöglicht es in Echtzeit 6-DOF Informationen zu ermitteln und diese während der Annäherung im Nahbereich zur Navigation zur nutzen.

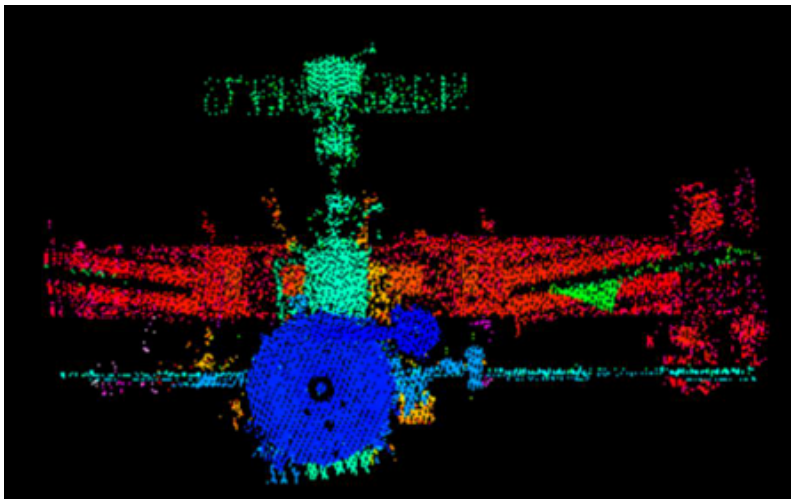


Abbildung A.1: RVS3000 Point Cloud Data der ISS, Entfernung ca. 30 meter [18]

Bei dem RVS3000-3D Sensor, handelt es sich um einen LIDAR Sensor welcher mittels eines nickelplattiertem AlBeMet Spiegels die Richtung des Lasers steuern kann, um somit innerhalb seines FOV's die Umgebung zu scannen [18]. Bei dem verbauten Laser handelt es sich um einen 1550nm Faser-Laser. Die an dem Ziel reflektierten Lichtwellen werden über die ADP-Unit detektiert. Über die TDOA-Methode (siehe Kapitel 2.1.1) können aus diesen Daten die Entfernungen ermittelt werden. Die hieraus generierten Daten können mit einem CAD-Modell abgeglichen werden, um die Auflösung der Daten zu verbessern [18]. Des weiteren können die Scan-Geschwindigkeiten, sowie das FOV (von 1° × 1°, bis 40° × 40°) angepasst werden um die Auflösung des Scans innerhalb der FOV's zu verbessern [18].

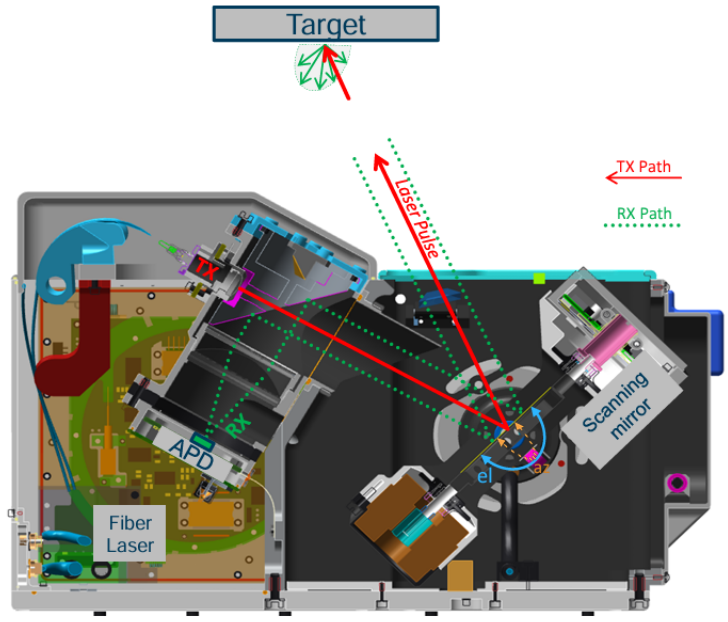


Abbildung A.2: Gezeigt ist der Aufbau des RVS3000(-3D) Systems, *el* und *az* stellen hierbei die Elevation und den Azimuth des FOV dar [18]

Zusammenfassend stellen Die RSV3000 und RVS3000-3D LIDAR-Systeme dar, welche in der Lage sind autonom kooperative und nichtkooperative Ziele zu erfassen und zu verfolgen. Die hieraus generierten Daten können unter anderem zur Manöverplanung für Rendezvous und Docking Missionen verwendet werden.

A.1.2 VGS/AVGS/NGAVGS

Ein weiteres System, welches von der NASA entwickelt wurde, stellt das NGAVAS dar. Bei diesem Sensorsystem handelt es sich um ein „...long-range proximity operations and docking sensor for use in an Automated Rendezvous and Docking (AR&D) system.“ [14, S. 1-2]. Dieses baut direkt auf dem VGS (Video Guidance System) und dem AVGS (Advanced Video Guidance System) auf, welche bereits ihre Einsatzfähigkeit in CEV's und COTS Systemen bestätigt haben [14, S. 1]. Ziel des NGAVGS ist es unter anderem Rendezvous, Nahbereichsmanöver und Fluidtransfers zu automatisieren.

Das VGS wurde ursprünglich entwickelt um Raumfahrzeuge die letzten 100m einer AR&D Mission zu führen. Das für den Bereich außerhalb dieser Entfernung verwendete Lokalisierungssystem nutzt GPS-basierte Daten [14, S. 2]. Das VGS wurde im Space Shuttle auf den Missionen STS-87 und STS-95 getestet, um ein Ziel auf dem SPARTAN free-flyer zu verfolgen. Die auf dieser Mission tatsächlich erfassten Einsatzbereiche des Sensors beliefen sich intermittierend auf 200m und konsistent auf 150m-10m [14, S. 2]. Da es bei diesen Missionen trotz vollständiger Absolvierung dieser zu Problemen in der Kontinuität des „tracking“-Vorgangs kam, entstand die Notwendigkeit einer Weiterentwicklung des Sensorsystems.

Das AVGS, welches die nächste Generation des VGS darstellt, war in der Lage Ziele in einer Reichweite von 300m bis 4m zu erfassen und zu verfolgen [14, S. 2]. Des Weiteren wurde ein „Spot Mode“ implementiert, welcher Bearing Informationen in einer Reichweite von bis zu 2 Kilometern realisieren konnte [14, S.2].

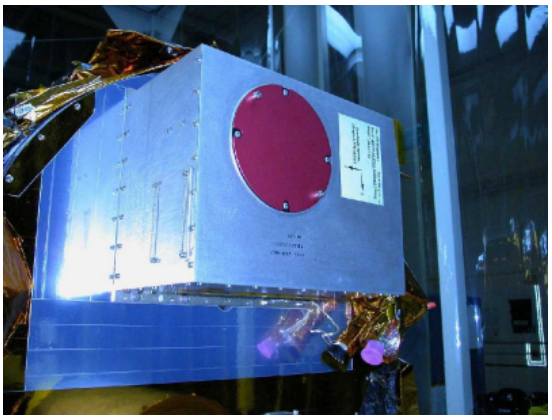


Abbildung A.3: Gezeigt ist der Aufbau des AVGS auf dem DART-System der NASA [14]

Das NGAVGS stellt die nächste und letzte Iteration des VGS's dar. Das NGAVGS beläuft sich auf einem Gewicht von 20lb, einem Volumen von 7 x 10 x 12 in. und einen durchschnittlichen

Energieverbrauch von 35W während der Benutzung und 14W im Standby Modus [14, S. 3]. Das Funktionsprinzip des NGAVGS basiert auf dem Verfahren der Laserbelichtung. Bei dieser werden zwei Aufnahmen generiert, welche mit unterschiedlichen Wellenlängen (806nm und 845nm) erstellt werden. Diese Aufnahmen werden voneinander subtrahiert um Punkte aus den belichteten Pixeln zu erzeugen. Darauf folgend werden diese mit der Zielgeometrie verglichen um die relativen Positionsvektoren und Attitude-Informationen zu generieren [14, S. 3]. Die Akquirierung der Bilddaten werden während des Verfolgens mittels eines DSP's realisiert, welcher ein Kommando an einen FPGA sendet um den nächsten Belichtungszyklus zu starten. Die generierten Bilddaten werden vom FPGA extern gespeichert und nach zwei Belichtungen ausgewertet. Die Unterschiede in den von einander subtrahierten Bilddaten entsteht hauptsächlich durch das von der Wellenlänge abhängige Reflexionsvermögen der Oberflächen des Ziel. Nachdem die Bilddaten vom FPGA verglichen wurden, werden diese zum „Patern matching“ an den DSP weiter gegeben Howard u. a. [14, S. 3].

(a) Background Image **(b)** Foreground Image

Abbildung A.4: Beispiel für Belichtungen bei unterschiedlichen Wellenlängen [14]

Abbildung A.5: Gezeigt ist das Resultat aus der Subtrahierung beider Bilddaten [14]

Das NGAVGS ist für Ziele ausgelegt welche über Retroreflektoren verfügen und stellt somit eine Sensortechnologie für kooperative Ziele dar. Das Sensorsystem verfügt über drei Betriebsmodi:„1) Standby (in which the sensor sends out status messages while awaiting further commands), 2) Acquisition (in which the sensor is commanded to actively seek a target and go into tracking once a valid target is found), and 3) tracking (the sensor is actively tracking a target that was found during Acquisition)“ [14, S. 5]. Mittels dieser Konfiguration ist das NGAVGS

in der Lage zuverlässlich Ziele in einer Reichweite von 5m-300m unter nicht näher spezifizierten Roll-, Nick- und Gierwinkeln zu verfolgen. Des Weiteren wurden Tests in Entfernungsbereichen von 2Km bis 5Km mit angepassten Retroreflektoren durchgeführt. Diese Tests zeigten aufgrund von Atmosphärischervarianz und -Störung sehr große Unsicherheiten in der Zuverlässigkeit und Reproduzierbarkeit der Messergebnisse [14, S.6].

A.1.3 Dragon Eye

Das Dragon Eye Sensorsystem ist Teil des aktuell verwendeten Sensorsystem der Dragon 2 des Unternehmens Space X. Die frei Verfügbaren Informationen zu diesen Sensorsystem sind auf Grund der Unternehmenspolitik von Space X nur schwer zugänglich, weswegen dieser Teil nur eine grobe Übersicht bezüglich der verwendeten Technologie gibt.

SpaceX verwendet in seinem „Dragon Capsule“ Transport Vehikel unter anderem das „Dragon Eye“ Sensorsystem für „guidance, navigation and controll“ Probleme [12]. Innerhalb dieses Systems, kommen LIDAR-Sensoren zum Einsatz welche drei dimensionale Image-Daten erzeugen um range und bearing Informationen relative zwischen der „Dragon Capsule“ und der ISS zu generieren[43, S.27].

Zudem kommen im „Dragon Eye“ Sensorsystem IR-Kameras zum Einsatz um das LIDAR-System beim „long range RPOD“ zu unterstützen und eine FDIR zu ermöglichen [12, S.2].

A.1.4 LIRIS

LIRIS wurde als Test neuer Rendezvous-Technologien in der fünften Version des ATV's der ESA bei der Annäherung an die ISS getestet [27, S.1]. Beim LIRIS-System handelt es sich um eine Kombination aus zwei IR-Kameras, einer Farbkamera und einem LIDAR-System [27, S.2]. Wobei das IR-Kamerasystem für Fernbereichsdetektion des Ziels eingesetzt wird und das LIDAR-System für den Nahbereich [27, S.2].

Hierbei wurden für verschiedene Entfernungsbereiche Anforderungen an den Informationsgewinn aus dem LIRIS-Modul definiert, wie der Abbildung A.6 zu entnehmen ist

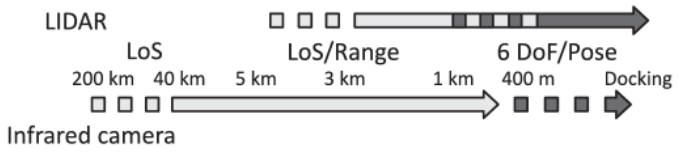


Abbildung A.6: Anforderungen an das LIRIS-System bei verschiedenen Distanzen [27]

Der Abbildung ist zu Entnehmen, dass eine aktive Distanzmessung erst ab einer Entfernung von 40 Km mittels der IR-Kamera vorgesehen ist. Die Anforderung in größeren Entfernungen definiert sich nur durch ein erkennen, sowie das Herstellen einer Sichtlinie zum Ziel.

Die Daten welche während der Mission generiert wurden, wurden während des Einsatzes zwischengespeichert und im Anschluss zur Auswertung herunter geladen. Nach Analyse der Sensordaten ergaben sich folgende Verwendungsmodelle für die Sensordaten.

Die Daten der individuellen Sensoren können verwendet werden um relative Navigationsaufgaben ohne nähere Informationen von der ISS außer Geometriedaten zu verwenden [27, S.29]. Des Weiteren können die Sensordaten verwendet werden um die Performance im Falle eines nichtkoorperativen Ziels zu evaluieren.

Die Analyse bezüglich der Verwendbarkeit der Daten hat gezeigt, dass für die IR-Kamera die Größe der ISS, bei einer Entfernung von 70km im FOV der Kamera, ungefähr 1 Pixel umfasst [27, S.32]. Auch wenn die Detektion während der Mission erfolgreich verlief bleibt zur weiteren Untersuchung offen, ob die Detektion auch ohne Ambiguität möglich ist [27, S.32]. Hierbei stellt sich vor allem die Frage nach dem Dynamikumfang der Kamera und dem daraus entstehenden Signal/Rausch- Verhältnis in größeren Distanzen und variierenden Belichtungsverhältnissen [27, S.32].

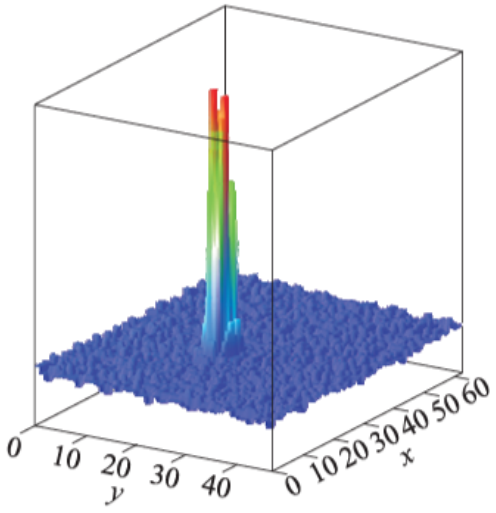


Abbildung A.7: IR-Kameradaten im Vergleich zum Hintergrundrauschen bei 9Km Entfernung zur ISS [27]

Bei Analyse der LIDAR-Daten wurden die für den geplanten Einsatz im Mittel- und Nahbereich generierten Daten mit den vom ATV generierten Trajektoriedaten verglichen, um die Güte der durch die Sensordaten erzielten Entfernungsvorhersagen zu überprüfen [27, S.33]. Ergebnis dieser Analyse war, dass ein Großteil der Messungen eine Varianz von 2-4m von der Referenztrajektorie aufweist (mit Maximalabweichungen von 6m) [27, S.32]. Somit befinden sich die Messdaten in-

nerhalb des Genauigkeitsbereiches der Referenztrajektorie [27, S.32]. Die geplotteten Ergebnisse der Messdaten ist Abbildung A.8 zu entnehmen.

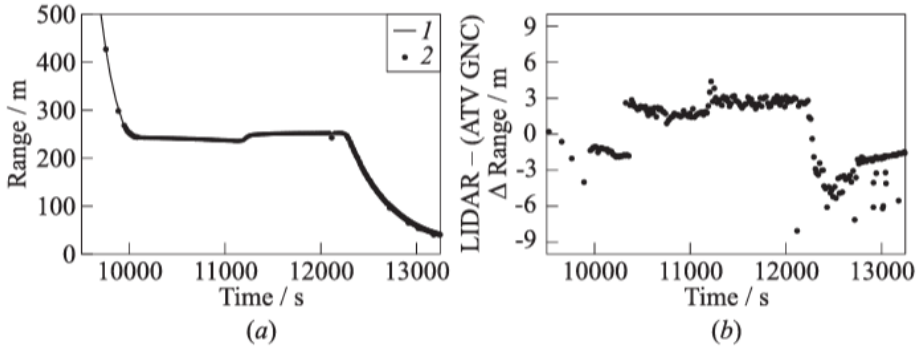
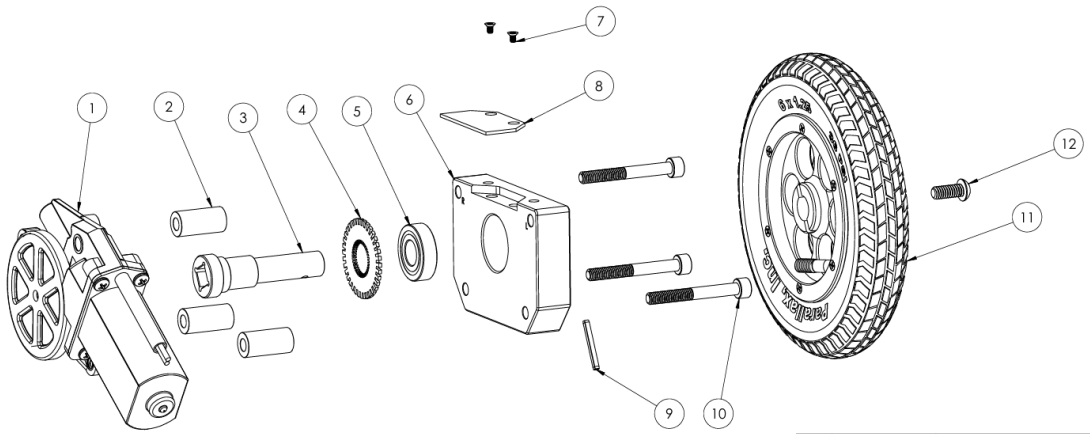


Abbildung A.8: Vergleich der (a) ATV Referenztrajektorie und (b) LIDAR-Daten [27]

A.2 Explosionszeichnung des Antriebssystems

Dieser Teil des Anhangs dient der Beilegung der technischen Zeichnung, welche im Zuge dieser Arbeit als Referenzen zur Anwendung kamen. Eine genaue Auflistung dieser ist den nächsten Unterkapiteln zu entnehmen.



ITEM NO.	PartNo	DESCRIPTION	QTY.
1	750-90006	Motor, 12 VDC (Right)	1
2	713-00030	Spacer, 0.50" x 1", RND, AL	3
3	34031	Robot Drive Axle 2	1
4		Encoder Disk	1
5	724-00001	Ball Bearing, 0.5" ID x 1.125" OD	1
7		Screw, 4-40 x 3/16" FH, PH, ZN	2
8	29321	Quardrature Encoder	1
9	700-00100	1/8" x 1" Roll Pin	1
10	710-00031	Screw M6, 1mm x 50mm SH, Black	3
11		Assy Wheel Rim, w/tube	1
12	710-00105	Screw, 1/4-20 x 5/8", BH, SS	1

Abbildung A.9: Explosionszeichnung des Antriebsystems [36]

B Software Code

B.1 Softwarecode zur Datenauswertung

Das folgende Kapitel dient der Auflistung des in dieser Abschlussarbeit verwendeten Softwarecodes zur Datenauswertung. Der ROS2 Softwarecode ist dem digitalen Anhang zu entnehmen, da dieser zu Umfangreich für den Rahmen der Abschlussarbeit ist.

Scan-Datenauswertung der Nullmessung

Listing B.1: Pythoncode zur Auswertung LIDAR Nullmessung

```
1  import pandas as pd
2  import os
3
4  dateipfad = r'C:\Users\Seanh\Desktop\Masterarbeit\software\Daten\scan_0messung.
       csv'
5  #einlesen der CSV
6  df = pd.read_csv(dateipfad)
7
8  #Berechnung der Mittelwert der Spalten 11 - 139
9  mittelwerte_der_spalten = df.iloc[:, 10:138].mean()
10
11 #Berechnung des Mittelwerts der Mittelwerte
12 gesamtmittelwert = mittelwerte_der_spalten.mean()
13
14 #Standardabweichung der Mittelwerte der Spalten
15 gesamtstd = mittelwerte_der_spalten.std()
16 #Ausgabe der Ergebnisse
17 print("Gesamtmittelwert =",gesamtmittelwert)
18 print("Gesamtstd =",gesamtstd)
```

Datenauswertung

Listing B.2: Pythoncode zum benchmarken des LIDAR-Sensors

```
import pandas as pd
import os

dateipfad = r'C:\Users\Seanh\Desktop\Masterarbeit\software\Daten\scan0_05m.csv'
#Umwandlung der .csv in einen Dataframe
df = pd.read_csv(dateipfad)
#erstellen der Liste der Minimalwerte
minimalwerte = []


for index, row in df.iterrows():
    #Auswahl der LIDAR-Daten aus \scan Nachricht
    filtered_values = row.iloc[10:138]
    #Filterung der Daten um Messfehler zu eleminieren
    filtered_values = filtered_values[(filtered_values > 0.01) & (
        filtered_values < 0.1)]

    #Auffuellen und Generieren der Minimalwertliste
    if len(filtered_values) > 0:
        #Finden des Minimalwertes der gefilterten Daten
        min_value = filtered_values.min()
        minimalwerte.append(min_value)
    else:
        #erstellen von nan Werten falls keine Werte vorhanden sind,  um den
            Mittelwert nicht zu beeinflussen
        minimalwerte.append(float('nan'))

#Umwandeln der Datenstruktur von List in Series um Mittelwert und STD zu
    berechnen
minimalwerte_series = pd.Series(minimalwerte)

#Berechnen des Mittelwerts und der Standardabweichung
mittelwert_minimalwerte = minimalwerte_series.mean()
standardabweichung_minimalwerte = minimalwerte_series.std()

#Ausgabe der Werte
print(f"Mittelwert = {mittelwert_minimalwerte}")
print(f"Standardabweichung = {standardabweichung_minimalwerte}")
```

Auswertung der Varianzen über die Zeit

Listing B.3: Pythoncode zur Auswertung der Kovarianzmatrix über die Zeit

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#Dateipfad der ROS2 pose_daten
dateipfad = r'C:\Users\Seanh\Desktop\Masterarbeit\software\Daten\pose daten\
    pose_05m.csv'

df = pd.read_csv(dateipfad)

#Hauptdiagonalwerte der Kovarianzmatrix
cov_diagonal_values = []

#for Schleife um die Hauptdiagonalwerte der Kovarianzmatrix jeder Spalte
    abzuspeichern
for index, row in df.iterrows():
    #Kovarianzmatrix befindet sich in den letzten 36 Eintraegen jeder Spalte
    covariance = np.array(row[-36:])
    #Extraktion der Hauptdiagonalwerte aus Reihe
    diag_values = covariance[[0, 7, 14, 21, 28, 35]]

    #appenden der Liste aller Hauptdiagonalwerte
    cov_diagonal_values.append(diag_values)

#Umwandlung der Liste in ein DataFrame mit zugehoerigen Ueberschriften
cov_diagonal_df = pd.DataFrame(cov_diagonal_values, columns=['cov_x', 'cov_y', '
    cov_z', 'cov_roll', 'cov_pitch', 'cov_yaw'])

#erstellen der Zeitintervalle in Sekunden
time_step = 0.16
time_axis = np.arange(0, len(cov_diagonal_df) * time_step, time_step)
#Plotten der Hauptdiagonalwerte ueber die Zeit
plt.figure(figsize=(10, 6))

#Plot der einzelnen Kovarianzen ueber die Zeit
plt.plot(time_axis, cov_diagonal_df['cov_x'], label='cov_x', color='r')
plt.plot(time_axis, cov_diagonal_df['cov_y'], label='cov_y', color='g')
plt.plot(time_axis, cov_diagonal_df['cov_z'], label='cov_z', color='b')
plt.plot(time_axis, cov_diagonal_df['cov_roll'], label='cov_roll', color='c')
plt.plot(time_axis, cov_diagonal_df['cov_pitch'], label='cov_pitch', color='m')
plt.plot(time_axis, cov_diagonal_df['cov_yaw'], label='cov_yaw', color='y')

#Plotparameter
plt.xlabel('Versuchszeit (in Sekunden)')
plt.ylabel('Kovarianz')
plt.title('Hauptdiagonalwerte der Kovarianzmatrix ueber die Zeit')
plt.legend()

```

```
45 #Plot anzeigen
46 plt.grid(True)
47 plt.show()
```

Abweichung der Koordinatensysteme

Listing B.4: Pythoncode zur Bestimmung der Abweichung der Koordinatensysteme nach durchfahren des Parkours

```
1  import numpy as np
2
3  def quaternion_to_rotation_matrix(q):
4
5      w, x, y, z = q
6
7      #Berechnung der Rotationsmatrix aus Quaternioneintraegen
8      R = np.array([
9          [1 - 2*(y**2 + z**2), 2*(x*y - z*w), 2*(x*z + y*w)],
10         [2*(x*y + z*w), 1 - 2*(x**2 + z**2), 2*(y*z - x*w)],
11         [2*(x*z - y*w), 2*(y*z + x*w), 1 - 2*(x**2 + y**2)]
12     ])
13
14     return R
15
16 #hier Quaternion der letzten Transformation einfuegen
17 q = [0.8813473241615482, 0, 0, 0.47246893463303913]
18
19 #Berechnung der Rotationsmatrix zum Quaternion q
20 rotation_matrix = quaternion_to_rotation_matrix(q)
21 #translatorische Verschiebung der Koordinatensysteme odom und base_link
22 odom_baselink = np.array([-0.07168241319740412,2.5368459820929528, 0])
23
24
25 #translatorische Verschiebung der Koordinatensysteme map und odom
26 map_odom = np.array([2.156360289762252,-1.3565499456115564, 0])
27 #Produkt der Transformationsmatrix und des Verschiebungsvektors odom zu baselink
28 odom_baselink_in_map_frame = rotation_matrix @ odom_baselink
29
30 #Berechnung der translatorischen Abweichung zwischen den Koordinatensystemen map
       und base_link
31 abweichung_2 =  odom_baselink_in_map_frame + map_odom
32
33 print("Abweichung_2 =",abweichung_2)
34 print(rotation_matrix)
```