

Fachhochschule Aachen  
Fachbereich Elektrotechnik und Informationstechnik  
Ingenieur-Informatik

Bachelorarbeit

# Range-Only Simultaneous Localization and Mapping mittels Ultra-Wideband

Albert Kasdorf  
geb. am 29.12.1984 in Pawlodar  
Matr.-Nr.: 3029294

Gutachter:  
Prof. Dr. rer. nat. Alexander Ferrein  
Dr. Stefan Schiffer



---

## **Eidesstattliche Erklärung**

---

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe. Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

---

Ort, Datum

---

Albert Kasdorf



---

# Inhaltsverzeichnis

---

<b>Abbildungsverzeichnis</b>	<b>vii</b>
<b>Tabellenverzeichnis</b>	<b>ix</b>
<b>1. Einführung</b>	<b>1</b>
1.1. Zielsetzung . . . . .	2
1.2. Gliederung . . . . .	2
<b>2. Grundlagen</b>	<b>5</b>
2.1. Verfahren für die Entfernungsbestimmung . . . . .	5
2.1.1. Single-sided Two-way Ranging . . . . .	6
2.1.2. Double-sided Two-way Ranging . . . . .	7
2.2. Geometrie . . . . .	8
2.2.1. Gleichseitiges Dreieck . . . . .	8
2.2.2. Regelmäßiges Fünfeck . . . . .	8
2.3. Wahrscheinlichkeitstheorie . . . . .	9
2.4. Zustandsschätzer . . . . .	10
2.4.1. Bayes Filter . . . . .	10
2.4.2. Kalman Filter . . . . .	11
2.4.3. Extended Kalman Filter . . . . .	12
2.5. Partikel Filter . . . . .	12
2.5.1. Monte Carlo Lokalisierung . . . . .	15
2.6. Simultaneous Localization and Mapping . . . . .	15
2.6.1. EKF-SLAM . . . . .	16
2.6.2. FastSLAM . . . . .	16
2.7. Robot Operating System . . . . .	19
<b>3. Stand der Forschung und Technik</b>	<b>21</b>
3.1. Preliminary results in range-only localization and mapping . . . . .	21
3.2. Robust range-only beacon localization . . . . .	23
3.3. A pure probabilistic approach to range-only SLAM . . . . .	23
3.4. Efficient probabilistic range-only SLAM . . . . .	24
3.5. A robust method of localization and mapping using only range . . . . .	25

<b>4. Ultra-Wideband</b>	<b>27</b>
4.1. Historie . . . . .	27
4.2. Erstellte Hardware . . . . .	29
4.2.1. Anforderungen . . . . .	29
4.2.2. Hardware Zusammenstellung . . . . .	29
4.2.3. Prototypen . . . . .	32
4.2.4. Platinendesign . . . . .	33
4.2.5. Steuersoftware . . . . .	35
4.2.6. Kalibrierung . . . . .	37
<b>5. Umsetzung des RO-SLAM in ROS</b>	<b>43</b>
5.1. Roboterplattform . . . . .	43
5.2. Softwarearchitektur . . . . .	44
5.2.1. ROS Hauptmodule . . . . .	45
5.2.2. ROS Hilfsmodule . . . . .	47
5.2.3. MRPT . . . . .	48
<b>6. Evaluation</b>	<b>51</b>
6.1. Batterielaufzeit . . . . .	51
6.2. Kalibrierung . . . . .	51
6.3. Entfernungsmessung . . . . .	54
6.4. Trajektorie . . . . .	56
6.5. RO-SLAM . . . . .	58
6.5.1. Positionsschätzung . . . . .	58
6.5.2. Positionsschätzung der UWB Module . . . . .	58
6.5.3. Konvergenz der WDF . . . . .	61
<b>7. Zusammenfassung und Ausblick</b>	<b>63</b>
7.1. Ausblick . . . . .	66
7.2. Fazit . . . . .	67
<b>Literaturverzeichnis</b>	<b>69</b>
Internetquellen . . . . .	70
Handbücher . . . . .	71
<b>Anhang</b>	<b>73</b>
<b>Anhang A. Abbildungen</b>	<b>75</b>
<b>Anhang B. Tabellen</b>	<b>79</b>

---

# Abbildungsverzeichnis

---

2.1.	Ablauf des Single-sided Two-way Ranging Verfahrens. . . . .	6
2.2.	Ablauf des Double-sided Two-way Ranging Verfahrens. . . . .	7
2.3.	Ablauf des Double-sided Two-way Ranging Verfahren mit einem Tag und drei Ankern. . . . .	8
2.4.	Lineare und nicht lineare Transformation einer normalverteilten Zufallsvariable. . . . .	13
2.5.	Darstellung einer Zufallsvariable vor und nach der Abbildung, durch eine nicht lineare Funktion, mittels einem Partikel Filter. . . . .	14
2.6.	Darstellung des SLAM Problems als bayessches Netz. . . . .	18
2.7.	Representation der Partikel in FastSLAM. . . . .	18
2.8.	Zuordnung der Landmarken zu den Wahrnehmungen pro Pfadhypothese. . . . .	18
3.1.	Darstellung der Beacons in einem Probability Grid. . . . .	22
3.2.	Annäherung einer ringförmigen Verteilung an eine Normalverteilung. . . . .	22
3.3.	Entwicklung eines Hilfspartikel Filters zu einem EKF. . . . .	24
3.4.	Modellierung der ringförmigen Verteilung über radial angeordnete Normalverteilungen. . . . .	24
3.5.	Darstellung der tatsächlichen und angenäherten Verteilung in Kartesischen und Polarkoordinaten. . . . .	26
4.1.	DecaWave IC Pin Belegung . . . . .	30
4.2.	Adafruit Pro Trinket . . . . .	31
4.3.	Adafruit Pro Trinket LiPoly/LiIon Backpack . . . . .	31
4.4.	Schaltplan des UWB Moduls. . . . .	33
4.5.	Die ersten zwei Prototypen der UWB Module. . . . .	33
4.6.	Designstufen eines elektrischen Bauelementes. . . . .	34
4.7.	Fertige UWB Module. . . . .	35
4.8.	Antennenverzögerung zwischen zwei UWB Modulen. . . . .	40
5.1.	Die Frontansicht der fertig aufgebauten Roboterplattform. . . . .	44
5.2.	Softwarearchitektur der ROS Module für den RO-SLAM. . . . .	45
5.3.	Statische Transformationen der Roboterplattform. . . . .	46
5.4.	Belegtheitskarte der Messstrecke. . . . .	47
5.5.	Trajektorie einer Messfahrt der Roboterplattform. . . . .	48

5.6. Positionsschätzung für die Roboterplattform mit roten Pfeilen und für die UWB Module mit blauen Pfeilen. . . . .	49
6.1. Versuchsaufbau für die Kalibrierung von drei UWB Modulen. . . . .	52
6.2. Histogramm und Wahrscheinlichkeitsdichtefunktion der kalibrierten Entfernungsmessungen. . . . .	53
6.3. Versuchsaufbau für die Kalibrierung von fünf UWB Modulen. . . . .	54
6.4. Tatsächliche und gemessene LoS- und NLoS Entfernungen. . . . .	55
6.5. Die Trajektorien der verschiedenen Odometriequellen. . . . .	57
6.6. Resultierende Trajektorie aus der Positionsschätzung des RO-SLAM mit realen und virtuellen UWB Modulen. . . . .	59
6.7. Positionsschätzungen der realen und virtuellen UWB Modulen zu bestimmten Zeitpunkten. . . . .	60
A.1. Der Ablauf der Discovery- und Range-Phase zwischen einem Tag und mehreren Ankern. . . . .	75
A.2. Ein gleichseitiges Dreieck. . . . .	76
A.3. Ein regelmäßiges Fünfeck in rot und ein Pentagramm in violett. . . . .	76
A.4. PCB-Adapterboard für den DWM1000. . . . .	76
A.5. Das Antennenabstrahlmuster in horizontaler Richtung. . . . .	77
A.6. Antennenfreiräume . . . . .	77
A.7. Versuchsaufbau für die NLoS-Entfernungsmessung. . . . .	78
A.8. Rechteckige Trajektorien der verschiedenen Odometriequellen. . . . .	78

---

# Tabellenverzeichnis

---

6.1.	Berechnete Werte für die Antennenverzögerung pro UWB Modul. . . . .	52
6.2.	Stochastische Eigenschaften der UWB Module ohne und mit Antennenkalibrierung bei einem Abstand von 1,73 m. . . . .	53
6.3.	Berechnete Werte für die Antennenverzögerung pro für das regelmäßige Fünfeck. .	54
6.4.	Stochastische Eigenschaften der Entferungen zwischen der Odometrie- und der Ground Truth Position. . . . .	57
6.5.	Entfernung der Positionsschätzung der realen UWB Module zu den Ground Truth Positionen. . . . .	61
6.6.	Entfernung der Positionsschätzung der virtuellen UWB Module zu den Ground Truth Positionen. . . . .	61
6.7.	Die Zeitspanne und die zurückgelegte Entfernung der Roboterplattform bis die WDF durch das EKF Verfahren abgebildet wird. . . . .	62
B.1.	Pinbelegung zwischen dem DWM1000 und Pro Trinket. . . . .	79
B.2.	Pinbelegung zwischen dem LiIon Backpack und dem Pro Trinket. . . . .	79
B.3.	Pinbelegung zwischen dem CP2104 Friend und dem Pro Trinket. . . . .	79
B.4.	Materialkosten pro Tag bzw. Anker. . . . .	80
B.5.	Stochastische Merkmale der LoS–Entfernungsmessung. . . . .	81
B.6.	Stochastische Merkmale der NLoS–Entfernungsmessung mit einem wassergefüllten Kunststoffbehälter. . . . .	81
B.7.	Stochastische Merkmale der NLoS–Entfernungsmessung mit einem Aluminiumblech in einer Entfernung von 50 cm. . . . .	82
B.8.	Stochastische Merkmale der NLoS–Entfernungsmessung mit einem Aluminiumblech in einer Entfernung von 5 cm. . . . .	82



# 1.

---

## Einführung

---

In der Zeit vor den Navigationsgeräten wurden auf deutschen Straßen noch regelmäßig faltbare Straßenkarten von den Beifahrern verwendet um den Fahrer den Weg zu weisen. Bevor eine Straßenkarte verwendet werden kann, muss diese erstellt werden. Dieser Prozess ist unter dem Begriff Kartenerstellung (engl. Mapping) bekannt. Der Detailgrad hängt dabei stark vom Verwendungszweck ab. Der erste Schritt nach dem entfalten der Straßenkarten bestand in der Lokalisierung (engl. Localization), also der Bestimmung der ungefähren Fahrzeugposition und dem Ziel der Reise auf der Straßenkarte. Darauf aufbauend wurde vom Beifahrer dann eine Route zwischen der aktuellen Fahrzeugposition und dem Ziel geplant und während der Fahrt weiterverfolgt, was auch als Pfad-Planung (engl. Path-Planning) bekannt ist.

Genauso wie der menschliche Agent, muss auch jeder mobile Roboter für sich die folgenden grundlegenden Fragen beantworten können: „Wo bin ich?“, „Wo bin ich bereits gewesen?“, „Wohin gehe ich?“ und „Welcher ist der beste Weg dahin?“. [1]

Außerhalb von geschlossenen Räumlichkeiten (engl. Outdoor) erfolgt die Lokalisierung in der Regel mittels Global Positioning System (GPS), unter der Voraussetzung das eine ungehinderte Verbindung (engl. Line of Sight (LoS)) zu den GPS Satelliten möglich ist. Die Lokalisierung ist in diesem Fall sehr einfach, da die GPS Koordinaten eindeutig sind und das Kartenmaterial bereits im gleichen Koordinatensystem vorliegt.

Innerhalb geschlossener Räumlichkeiten (engl. Indoor), wie in öffentlichen Gebäuden, Logistikhallen oder auch in Bergwerken, ist eine Lokalisierung mittels GPS nicht mehr möglich. Erschwerend kommt dazu, dass es in der Regel zu diesen Räumlichkeiten keine öffentlich verfügbaren Karten gibt oder diese sich wie im letzten Beispiel häufig ändern. Aus diesem Problemfeld haben sich Algorithmen für die Simultane Lokalisierung und Kartenerstellen (engl. Simultaneous Localization and Mapping (SLAM)) entwickelt.

Häufig werden SLAM Algorithmen verwendet um aus Kamerabildern oder 360° Abstandsmessungen eine Karte der Umgebung zu erstellen und sich in der Gleichen zu lokalisieren. Diese Sensoren setzen eine ungehinderte LoS voraus und reagieren sehr Empfindlich auf Verschmutzungen ihrer optischen Elemente. Sensoren deren Funktionsprinzip auf elektromagnetischen Wellen basieren, kommen auch ohne eine direkte Sichtverbindung (engl. Non-line of Sight (NLoS)) aus. Zu dieser Klasse von Sensoren gehört die Ultra-Wideband (UWB)-Technologie. Mit den hier in der Arbeit erstellten UWB Modulen ist nur eine Entfernungsmessung untereinander möglich. Diese beschränkten Informationen reichen jedoch für einen reinen entfernungsbasierenden SLAM (engl. Range-Only Simultaneous Localization and

Mapping (RO-SLAM)) Algorithmus aus. Hierbei werden nur die Informationen der Eigenbewegung und die Entfernung zu mehreren, vorher unbekannten, UWB Modulen genutzt um sich selbst zu Lokalisieren und eine Karte mit den Positionen der UWB Module zu erstellen.

Die UWB Module die stationär befestigt sind, ihre Position also nicht ändern und als fixe Referenzpunkte angesehen werden können, werden im Folgenden nur noch als Anker bezeichnet. Der Gegenspieler zum Anker ist der Tag. Dieser befindet sich auf der Roboterplattform und kann sich dementsprechend durch den Raum bewegen.

## 1.1. Zielsetzung

Diese Arbeit spaltet sich in zwei Bereiche auf. Im ersten Teil geht es um die Erstellung der UWB Module. Hierfür müssen die passenden elektrischen Komponenten ausgesucht, Prototypen entworfen und erprobt werden, ein Platinen-Layout erstellt und im Anschluss die UWB Module zusammengebaut werden. Mit der fertigen Hardware werden dann im folgenden empirische Versuche durchgeführt um die Sensorcharakteristiken wie Streubreite, Varianz und das LoS-/NLoS-Verhalten zu bestimmen.

In dem darauf aufbauenden zweiten Teil wird eine Roboterplattform mit einem UWB Modul ausgerüstet und ein Versuchsgelände mit mehreren Anker präpariert. Mit einem Steuergerät wird die Roboterplattform mehrmals durch das Versuchsgelände gesteuert und während dessen alle Sensordaten aufgezeichnet. Im Nachgang wird dann der RO-SLAM Algorithmus aus dem Mobile Robot Programming Toolkit (MRPT) Framework mit den Aufzeichnungen erprobt. Untersucht werden dabei die Genauigkeit der Roboter Lokalisierung und das Erstellen einer Karte von den zuvor unbekannten Anker-Positionen.

## 1.2. Gliederung

Im Kapitel zwei wird das grundlegende Wissensfundament für den RO-SLAM gelegt. Angefangen bei der Fragestellung wie eine Entfernungsmeßung mit den UWB Modulen durchgeführt wird und welche unterschiedlichen Verfahren es für den Nachrichtenaustausch existieren. Weiter zu den Grundlagen der Wahrscheinlichkeitstheorie und den darauf aufbauenden Bayes Filter, Kalman Filter, Partikel Filter und SLAM Verfahren. Den Abschluss des Kapitels bilden dann ein kurzer Einblick in das Robot Operating System (ROS) mit seinen wichtigsten Bestandteilen.

Das dritte Kapitel beschäftigt sich mit dem aktuellen Stand der Forschung und Technik. Angefangen bei den ersten Versuchen einen RO-SLAM mit einem Kalman Filter und einer guten initialen Positionsschätzung der Anker, hin zu der Modellierung der radialen Verteilungsfunktion, für den Kalman Filter, in Polarkoordinaten. Es werden auch Erfahrungen aus eher unbekannten Bereichen wie der Tiefseeuntersuchung miteinbezogen.

Mit dem vierten Kapitel erfolgt eine Einführung in die UWB-Technik und ein kurzer Abriss über die Entstehungsgeschichte. Danach konzentriert sich der Abschnitt auf die verschiedenen Stufen der Entwicklung eines einsatzfähigen UWB Moduls. Angefangen bei der Anforderungs-erhebung, dem Schaltungsaufbau, dem Platinen-Layout und der Steuersoftware.

Im darauffolgenden Kapitel fünf wird zuerst die Roboterplattform mit ihren spezifischen Eigenschaften vorgestellt. Eingeschlossen sind hierbei die verbauten Sensoren und deren Konfiguration in ROS. Zu den Sensoren zählt die Odometrie, der 2D-Laser-Entfernungsmesser und das UWB Modul. Weiterhin werden die verwendeten Softwaremodule aus ROS und dem MRPT Framework diskutiert.

Die Versuchsaufbauen und die Ergebnisse werden im sechsten Kapitel vorgestellt und diskutiert. Zum einen wird die maximale Betriebsdauer eines UWB Modul im nicht optimierten Softwarestand untersucht. Danach werden die Sensorcharakteristiken der UWB Module in Bezug auf die Entfernungsmessungen untersucht. Zu Letzt findet eine Untersuchung des Gesamtsystems mit dem RO-SLAM Algorithmus statt. Die erreichte Genauigkeit wird dabei mit einem Ground Truth Modell verglichen und diskutiert.

Das letzte Kapitel sieben liefert eine Zusammenfassung über die Arbeit und wirft einen Ausblick in die Zukunft sowie die weiteren Möglichkeiten die sich aus dieser Arbeit ergeben.



# 2.

---

## Grundlagen

---

Die Abschnitte mit der Wahrscheinlichkeitstheorie, dem Zustandsschätzer, dem Partikel Filter und der Simultaneous Localization and Mapping orientieren sich an dem Buch *Probabilistic robotics* [2] ohne dieses in jedem Absatz zu zitieren.

### 2.1. Verfahren für die Entfernungsbestimmung

Bei der Triangulation werden die Winkel zwischen mehreren Referenzpunkten bestimmt und dann die dazu gehörige Entfernung mittels trigonometrischer Funktionen berechnet. Dieses Verfahren ist auch unter den Namen Angle of Arrival (AOA) bzw. Direction of Arrival (DOA) bekannt. Um eine genau Ortsbestimmung durchzuführen müssen die Winkel sehr genau bestimmt werden. Um das zu bewerkstelligen werden im Empfänger mehrere Antennen zu einem Feld (engl. Antenna Array) zusammengefasst. Jedoch ist diese Konstruktion sehr teuer und empfindlich für Mehrwegeempfang (engl. Multipath) bzw. Signalabschattungen. [3–5]

Im Gegensatz dazu werden bei der Trilateration die Entfernungen zwischen mehreren Referenzpunkten betrachtet. Es werden dabei die Verfahren Time of Arrival (ToA) und Time Difference of Arrival (TDoA) unterschieden. Bei dem ToA-Verfahren wird zuerst die Zeitdifferenz zwischen dem Senden und Empfangen eines Funksignals berechnet. Mittels der Signallaufzeit (engl. Time of Flight (ToF)) und der Ausbreitungsgeschwindigkeit des Funksignals kann die Entfernung berechnet werden. Die Ortsbestimmung erfolgt dann über die Schnittpunkte von drei Kreisen (2D) bzw. vier Kugeln (3D) miteinander. Um dieses Verfahren anwenden zu können, ist es erforderlich das das Funksignal mit einem Zeitstempel des Startzeitpunktes versehen ist. Daraus folgt aber auch, das die Zeit zwischen Sender und Empfänger sehr genau synchronisiert werden müssen um den Fehler möglich klein zu halten. Bei dem TDoA-Verfahren werden die Zeitdifferenz zwischen dem Empfang des Funksignals an mehreren Empfängern ausgewertet. Dies hat den großen Vorteil das nur noch die Zeit zwischen den Empfängern synchronisiert werden muss. [5, 6]

Neben der Triangulation und Trilateration besteht auch die Möglichkeit auf Grund der empfangenen Signalstärke (engl. Signal Strength (SS), Received Signal Strength (RSS) oder auch Received Signal Strength Indication (RSSI)) Rückschlüsse über die Entfernung zu ziehen. Dazu muss die ursprüngliche Signalstärke und die Ausbreitungscharakteristik der elektromagnetischen Welle in der spezifischen Umgebung bekannt sein. [3, 5]

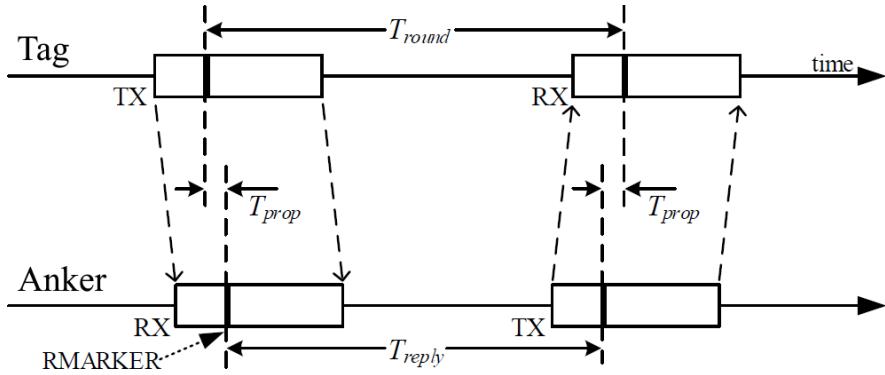


Abbildung 2.1.: Ablauf des Single-sided Two-way Ranging Verfahrens. [8]

In den nächsten zwei Abschnitten werden die DecaWave Entfernungsmessverfahren vorgestellt. Diese haben den Vorteil, dass keine Synchronisierung der Zeit zwischen Sender und Empfänger notwendig ist. Weiterhin besitzen die DecaWave UWB Transceiver zwei Eigenschaften, die die Entfernungsmessung ideal ergänzt. Zum einen wird jede erhaltene Nachricht mit einem lokalen Zeitstempel versehen, der über eine minimale Auflösung von ungefähr 15,65 ps verfügt. Hiermit wäre eine theoretische Ortsauflösung von ungefähr 5 mm möglich. Des Weiteren ist es möglich, den Sendezzeitpunkt einer Nachricht in die Zukunft zu legen. Damit lässt sich die Zeitspanne zwischen dem Empfang und der Antwort auf eine Nachricht im Voraus berechnen. Die Zeitspanne kann dann der Antwortnachricht als Nutzlast mitgegeben werden, um beim Empfänger die Umlaufzeit zu berechnen.

### 2.1.1. Single-sided Two-way Ranging

Das einfachste Verfahren, um aus der Umlaufzeit die Entfernung abzuschätzen, ist das Single-sided Two-way Ranging (SS-TWR) Verfahren. Dabei sendet der Tag eine Nachricht an den Anker und wartet ab, bis eine entsprechende Antwortnachricht eintrifft. Beide Module erhalten einen Zeitstempel für den Versand und Empfang von Nachrichten. Aus diesen kann dann die Antwort- ( $T_{reply}$ ) und Umlaufzeit ( $T_{round}$ ) berechnet werden, siehe Abbildung 2.1. [7, 8]

Die ungefähre ToF Zeitspanne ergibt sich aus der folgenden Gleichung:

$$T_{prop} = \frac{1}{2} (T_{round} - T_{reply}) \quad (2.1)$$

Damit der Tag die ToF Zeitspanne berechnen kann, benötigt er die  $T_{reply}$  Zeitspanne. Zu diesem Problem gibt es mehrere Lösungen. Die einfachste Lösung ist eine feste Antwortzeit die jedem Modul bekannt ist. Alternativ kann der Anker in der Antwortnachricht seine individuelle Antwortzeit übermitteln. Oder der Tag übermittelt dem Anker mit der initialen Nachricht wie lange der Anker warten muss bis er die Antwortnachricht verschickt. Hierbei muss die Zeitspanne groß genug gewählt sein, um den Anker die Möglichkeit zur Antwort zu lassen. Je nach Anforderung wird eine der vorherigen Methoden verwendet.

Der Nachteil bei diesem Verfahren besteht in dem Fehler der von der Antwortzeit abhängt. Bei Module verwenden zur Berechnung von  $T_{round}$  und  $T_{reply}$  ihre lokalen Zeitgeber. Beide

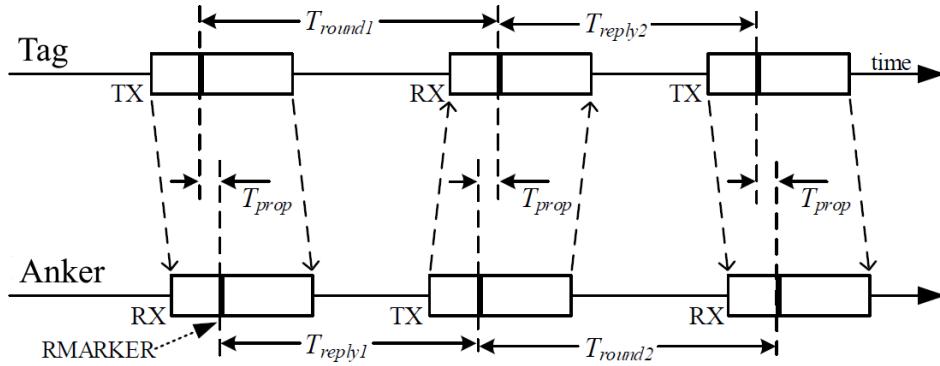


Abbildung 2.2.: Ablauf des Double-sided Two-way Ranging Verfahrens. [8]

Zeitgeber haben einen Offsetfehler  $e_A$  und  $e_B$  der von der Nennfrequenz abweicht. Die daraus abgeleitete ToF Zeitschätzung hat damit einen Fehler der mit der Antwortzeit wächst:

$$\text{error} \approx \frac{1}{2} (e_B - e_A) \times T_{reply} \quad (2.2)$$

### 2.1.2. Double-sided Two-way Ranging

Das Double-sided Two-way Ranging (DS-TWR) Verfahren stellt eine Verbesserung gegenüber dem SS-TWR Verfahren dar. Hierbei werden nun drei Nachrichten verwendet um jeweils die Umlaufzeiten zwischen Tag und Anker, und Anker und Tag zu berechnen, siehe Abbildung 2.2. Wenn die Umlaufzeit beim Tag berechnet werden soll, müssen die Zeitspannen  $T_{reply1}$  und  $T_{round2}$  zum Tag übermittelt werden. Für die letzte Zeitspanne erfolgt das mit einer vierten Nachricht die in dem Schaubild nicht abgebildet ist. [7, 8]

Die ungefähre ToF Zeitspanne ergibt sich aus der folgenden Gleichung:

$$T_{prop} = \frac{(T_{round1} \times T_{round2} - T_{reply1} \times T_{reply2})}{(T_{round1} + T_{round2} + T_{reply1} + T_{reply2})} \quad (2.3)$$

Der Fehler bei diesem Verfahren ergibt sich aus der folgenden Gleichung:

$$\text{error} = T_{prop} \times \left(1 - \frac{k_a + k_b}{2}\right) \quad (2.4)$$

Die Variablen  $k_a$  und  $k_b$  entsprechen hierbei den Offsetfehlern der Zeitgeber von der Nennfrequenz und liegen beide sehr nahe bei eins.

Mit diesem Verfahren ist es auch möglich, gleichzeitig die Entfernung zu mehr als einem Anker zu bestimmen, siehe Abbildung 2.3. Hierbei weist der Tag in der initialen Nachricht jedem Anker eine individuelle Antwortzeit zu. Danach wartet er bis alle Antwortnachrichten angekommen sind um im der letzten Nachricht jedem Anker seine Umlauf- und Antwortzeiten zu übermitteln. Jeder Anker kann nun individuell die ToF Zeitspanne für sich berechnen und dann dem Tag übermitteln. Diese letzte Nachricht ist auf der Abbildung 2.3 nicht aufgeführt.

In der Abbildung 2.3 wurde in der initialen Nachricht jedem Anker eine individuelle Antwortzeit zugeordnet. Woher wusste der Tag welche Anker vorhanden sind? Die Kommunikation zwischen Tag und Anker kann in zwei Phasen unterteilt werden, siehe Abbildung A.1. In der Discovery Phase schickt der Tag periodisch Blink-Nachrichten mit seiner

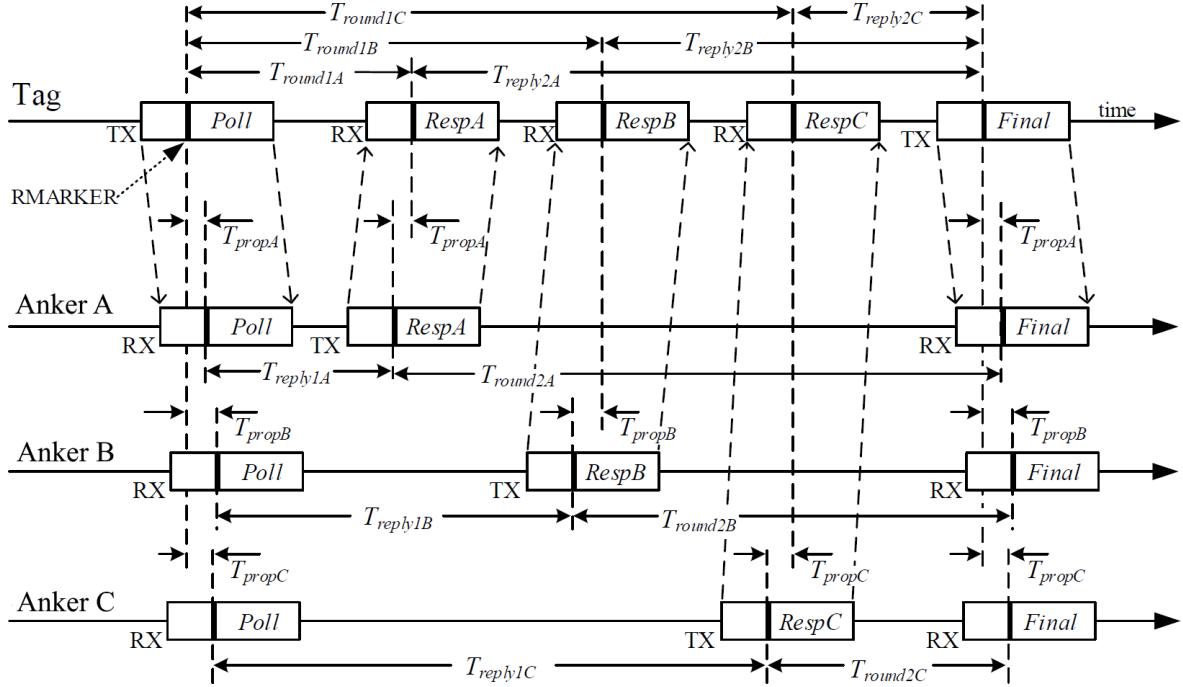


Abbildung 2.3.: Ablauf des Double-sided Two-way Ranging Verfahren mit einem Tag und drei Ankern. [8]

Identifikationsnummer an alle Module in der Umgebung. Empfängt ein Anker diese Nachricht, nimmt er den Tag in seiner Liste auf und übermittelt dem Tag eine Ranging Init Nachricht mit seiner Identifikationsnummer. Das Tag seinerseits nimmt den Anker in seiner Liste auf und weiß nun auch welche Anker er in der nächsten Entfernungsmessung berücksichtigen muss. Die Ranging Phase entspricht hierbei der bereits zuvor besprochenen Entfernungsmessung.

## 2.2. Geometrie

### 2.2.1. Gleichseitiges Dreieck

Ein gleichseitiges Dreieck zeichnet sich dadurch aus, dass alle drei Seiten gleich lang sind und jeder Innenwinkel einen Wert von  $60^\circ$  besitzt, siehe Abbildung A.2. Ist der Umkreisradius  $r_u$  eines gleichseitigen Dreiecks bekannt, so kann mit der Gleichung 2.5 auch die Seitenlänge  $a$  berechnet werden.

$$a = \frac{3}{\sqrt{3}} r_u \quad (2.5)$$

### 2.2.2. Regelmäßiges Fünfeck

Das regelmäßige Fünfeck zeichnet sich dadurch aus, dass alle fünf äußeren Seiten gleich lang sind und jeder Innenwinkel einen Wert von  $108^\circ$  besitzt, siehe Abbildung A.3. Aus der

Länge einer äußeren Seite  $a$  lässt sich zum einen der Umkreisradius  $r_u$ , siehe Gleichung 2.6, und die Diagonale  $d$  zwischen zwei Spitzen, siehe Gleichung 2.7, berechnen.

$$r_u = \frac{a}{10} \sqrt{50 + 10\sqrt{5}} \quad (2.6)$$

$$d = \frac{a}{2} (1 + \sqrt{5}) \quad (2.7)$$

## 2.3. Wahrscheinlichkeitstheorie

Ob sich ein Roboter an einer ganz bestimmten Position befindet, lässt sich in der Praxis nicht genau bestimmen. Auch durch das genauere Vermessen der Position bleibt eine gewisse Unsicherheit. Um diese Unsicherheit abzubilden, wird in der Robotik die Wahrscheinlichkeitstheorie angewendet. Zum Einsatz kommen hierbei kontinuierliche Zufallsvariablen und die mit ihnen verbundene Wahrscheinlichkeitsdichtefunktion (WDF). In der Regel wird eine Gaußsche Normalverteilung eingesetzt, die sich über den Mittelwert  $\mu$  und Varianz  $\sigma^2$  beschreiben lässt, siehe Gleichung 2.8. Häufig wird die Normalverteilung auch über  $\mathcal{N}(x; \mu, \sigma^2)$  abgekürzt.

$$p(x) = (2\pi\sigma^2)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right\} \quad (2.8)$$

Da es sich bei der Position nicht um einen skalaren Wert handelt, muss eine mehrdimensionale Normalverteilung eingesetzt werden, siehe Gleichung 2.9. Diese wird über einen Vektor der Mittelwerte  $\mu$  und eine symmetrische Kovarianzmatrix  $\Sigma$  beschrieben.

$$p(x) = \det(2\pi\Sigma)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right\} \quad (2.9)$$

Die multivariate Verteilung zweier Zufallsvariablen beschreibt die Wahrscheinlichkeit, mit welcher das Ereignis  $x$  sowie das Ereignis  $y$  auftritt. Sollten beide Zufallsvariablen unabhängig von einander sein, so lassen sich ihre Wahrscheinlichkeiten multiplizieren, siehe Gleichung 2.10.

$$p(x, y) = p(x)p(y) \quad (2.10)$$

Wenn eine Zufallsvariable, Wissen über eine andere Zufallsvariable besitzt, spricht man von der bedingten Wahrscheinlichkeit, siehe Gleichung 2.11.

$$p(x | y) = \frac{p(x, y)}{p(y)} \quad (2.11)$$

Die bedingte Wahrscheinlichkeit von zwei Zufallsvariablen die unabhängig von einander sind, reduzieren sich zu der Wahrscheinlichkeit einer Zufallsvariable, siehe Gleichung 2.10 und Gleichung 2.12.

$$p(x | y) = \frac{p(x)p(y)}{p(y)} = p(x) \quad (2.12)$$

Der Satz von Bayes sagt aus, das ein Verhältnis zwischen der bedingten Wahrscheinlichkeit  $p(x | y)$  und der umgekehrten Form  $p(y | x)$  besteht, siehe Gleichung 2.13.

$$p(x | y) = \frac{p(y | x) p(x)}{p(y)} \quad (2.13)$$

Der Nenner  $p(y)$  hängt nicht von  $x$  ab, ist daher für jedes Ereignis konstant und wird als Konstante  $\eta$  vor die Gleichung gesetzt, siehe Gleichung 2.14.

$$p(x | y) = \eta p(y | x) p(x) \quad (2.14)$$

## 2.4. Zustandsschätzer

Ein Roboter der sich in seiner Umwelt zurechtfinden soll, muss diese modellieren. Das Ergebnis der Modellierung bezeichnet man als Zustand. Ein Zustand beschreibt dabei alle Aspekte des Roboters und seiner Umwelt die einen Einfluss auf die Zukunft haben können. Ein Zustand kann dabei aus statischen sowie dynamischen Komponenten bestehen. Aus dem letzteren ergibt sich zwangsläufig, dass ein Zustand sich auch über die Zeit verändern kann. Ein Zustand kann z.B. die Pose des Roboters sein, die Positionen von stationären Landmarken oder aber auch der Ladezustand der Energieversorgung. Beschrieben wird er Zustand mit dem Symbol  $x$ .

Die meisten Roboter verfügen über Sensoren um Änderungen an Ihrer Umwelt wahrzunehmen. Diese Wahrnehmungen werden dabei mit dem Symbol  $z$  beschrieben. Zusätzlich kann der Roboter aktiv Änderungen an seiner Umwelt vornehmen, in dem er Steuerbefehle an die Aktorik sendet und sich somit durch seine Umwelt bewegt. Steuerbefehle werden mit dem Symbol  $u$  beschrieben.

Um Auszudrücken, dass die Information des Zustandes, der Wahrnehmung und der Steuerbefehle, zu einem bestimmten Zeitpunkt gehören wird der Index  $x_t$  verwendet. Eine Zeitspanne wird über den Index  $x_{1:t}$  ausgedrückt. Im mathematischen Sinne werden alle drei Größe als Spaltenvektoren beschrieben, die nicht über die gleiche Dimension verfügen müssen. Zum Beispiel könnte der Zustand aus den X-/Y-Position und der Orientierung  $\theta$  zusammengesetzt sein, während die Wahrnehmung aus vielen hunderten Entfernungsmessungen bestehen.

Der Zustand der Umwelt kann nicht direkt gemessen werden, stattdessen muss dieser aus den Daten der Wahrnehmung und der Steuerbefehle geschlussfolgert werden. Das interne Wissen des Roboters über den Zustand seiner Umwelt wird dabei als Belief  $bel(x_t)$  bezeichnet. Der Belief wird dabei durch eine bedingte Wahrscheinlichkeitsverteilung repräsentiert. Jeder Zustandshypothese ordnet der Belief dabei eine Wahrscheinlichkeit für ihre Gültigkeit zu.

### 2.4.1. Bayes Filter

Bei dem Bayes Filter Algorithmus handelt es sich um einen sehr abstrakte Beschreibung eines rekursiven Zustandsschätzer, siehe Algorithmus 2.1. Das heißt, aufbauend auf der aktuellen Zustandsschätzung wird mit dem eintreffen neuer Wahrnehmungs- oder Steuerbefehls-Daten eine neue Zustandschätzung generiert. Abhängig vom Vorwissen des initialen Zustandes  $bel(x_0)$

liefert diese entweder eine punktförmige Verteilung durch den initialen Zustand zurück oder eine stetige Gleichverteilung über den kompletten Zustandsraum.

```

1: Algorithm Bayes_Filter( $bel(x_{t-1})$ ,  $u_t$ ,  $z_t$ ):
2:   forall  $x_t$  do
3:      $\bar{bel}(x_t) = \int p(x_t | u_t, x_{t-1}) bel(x_{t-1}) dx_{t-1}$ 
4:      $bel(x_t) = \eta p(z_t | x_t) \bar{bel}(x_t)$ 
5:   end
6:   return  $bel(x_t)$ 
```

Algorithmus 2.1 : Bayes Filter Algorithmus [2]

Der Bayes Filter geht dabei in zwei Schritten vor. Im ersten Schritt wird eine Prognose (engl. Prediction) für den neuen Zustand erstellt, siehe Zeile 3. Der neue Zustand wird dabei aus dem vorherigen Zustand und dem aktuellen Steuerbefehl gebildet, siehe  $p(x_t | u_t, x_{t-1})$ . Die daraus resultierende Verteilung wird mit der vorherige Belief Verteilung  $bel(x_{t-1})$  multipliziert und ergibt die Prognose  $\bar{bel}(x_t)$  für den neuen Zustand. Im zweiten Schritt wird die erstellt Prognose mit der aktuellen Wahrnehmung korrigiert (engl. Correction), siehe Zeile 4. Hierbei ist zu beachten, das für jede Zustandshypothese  $x_t$  die erstelle Prognose  $\bar{bel}(x_t)$  mit der Wahrscheinlichkeit multipliziert wird, das die Wahrnehmung zu der Zustandprognose  $p(z_t | x_t)$  zutrifft.

### 2.4.2. Kalman Filter

Die Familie der Gauß Filter, zu denen auch der Kalman Filter gehört, ist eine der frühesten konkreten Implementierungen des Bayes Filters. Die Idee hinter jedem Gauß Filter liegt dabei, das der Belief durch eine mehrdimensionale Normalverteilung repräsentiert wird, die eindeutig durch die ersten beiden Momente Mittelwert  $\mu$  und Kovarianz  $\Sigma$  beschrieben wird. Der große Vorteil einer normalverteilten Zufallsvariable ist, dass das Ergebnis einer Lineartransformation mit einer anderen normalverteilten Zufallsvariable wieder zu einer normalverteilten Zufallsvariable führt. Zu den Nachteilen zählt, dass der Gauß Filter nur im kontinuierlichen Zustandsraum angewendet werden kann und durch die unimodale Verteilung nur für die Positionsverfolgung (engl. Position Tracking Problem) in Frage kommt.

Genauso wie der Bayes Filter geht auch der Kalman Filter in zwei Schritten vor, siehe Algorithmus 2.2. Im ersten Schritt wird aus dem vorherigen Zustand  $\mu_{t-1}$  und  $\Sigma_{t-1}$  und den aktuellen Steuerbefehlen  $u_t$  die Prognose für den neuen Zustand berechnet, siehe Zeile 2–3. Über die Matrix  $A_t$  wird festgelegt, wie sich der Zustand von  $t - 1$  nach  $t$  entwickelt, ohne die Einwirkungen der Steuerbefehle oder von Rauschen. Die Matrix  $B_t$  beschreibt wie die Steuerbefehle die Zustandsänderung von  $t - 1$  nach  $t$  beeinflussen. Im zweiten Schritt wird zuerst der Kalman Gain  $K_t$  berechnet, siehe Zeile 4. Bei dem Kalman Gain handelt es sich um ein Gewicht, das entscheidet wie stark die Prognose mit der Wahrnehmung korrigiert wird, siehe Zeile 5–6. Mit der Matrix  $C_t$  wird definiert wir die Wahrnehmungen auf den Zustand abgebildet werden. Die beiden Matrizen  $R_t$  und  $Q_t$  bilden dabei das unabhängige und zufällig normalverteilte Rauschen ab.

```

1: Algorithm Kalman_Filter( $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$ ):
2:    $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$ 
3:    $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$ 
4:    $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$ 
5:    $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$ 
6:    $\Sigma_t = (I - K_t C_t) \bar{\Sigma}_t$ 
7: return  $\mu_t, \Sigma_t$ 

```

Algorithmus 2.2 : Kalman Filter Algorithmus [2]

### 2.4.3. Extended Kalman Filter

Der Kalman Filter geht von der Annahme aus, dass die Zustandsänderung und die Wahrnehmungen durch eine lineare Funktion beschrieben werden können. Diese Annahme ist entscheidend für das ordnungsgemäße Funktionieren des Kalman Filters. In der Abbildung 2.4a ist im rechten unteren Bereich die Zufallsvariable  $x$  mit ihrer Normalverteilung dargestellt. Wird diese über eine lineare Funktion  $y = ax + b$  auf die Zufallsvariable  $y$  abgebildet. Es ist klar zu erkennen, dass die Normalverteilung für die Zufallsvariable  $y$  erhalten bleibt.

Das Hauptproblem bei der linearen Annahme ist, dass viele Zustandsänderung und Wahrnehmungen nicht mit einer linearen Funktion beschrieben werden können, z.B. rotatorische Zustandsänderungen. Die Abbildung 2.4b verdeutlicht, was mit der Normalverteilung einer Zufallsvariable  $x$  passiert, wenn diese durch eine nicht lineare Funktion  $g(x)$  abgebildet wird. Von der Normalverteilung der Zufallsvariable  $y$  bleibt nicht mehr viel übrig.

Der Extended Kalman Filter (EKF) löst das Problem in dem eine Linearisierung der nicht linearen Funktion vorgenommen wird, siehe Abbildung 2.4c. Hierzu wird eine Taylorreihe des ersten Grades gebildet. Der Funktionswert der Mittelwertes  $g(\mu)$  dient dabei als Entwicklungsstelle. Durch die Linearisierung wird die Normalverteilung der Zufallsvariable  $x$  wieder ordnungsgemäß auf die Zufallsvariable  $y$  abgebildet. Da es sich bei der Linearisierung nicht um die Originalfunktion handelt entsteht ein Linearisierungsfehler der am Unterschied zwischen der durchgezogenen und gestrichelten Kurve zu erkennen ist, siehe Abbildung 2.4.

## 2.5. Partikel Filter

Bisherige Filter haben eine parametrisierte Form genutzt, um die Verteilung zu modellieren. Der Partikel Filter geht einen anderen Weg und verwendet stattdessen eine Menge von Stichproben (engl. Samples) aus der gewünschten Verteilung. Dadurch ist es möglich multimodale Verteilungen abzubilden, die jedoch nur einer Annäherung entsprechen. Weiterhin ist es möglich Zufallsvariablen durch eine nicht lineare Funktion abzubilden, siehe Abbildung 2.5. Im unteren rechten Bild wird eine Menge von Stichproben aus einer Normalverteilung dargestellt. Diese werden dann durch die nicht lineare Funktion  $g(x)$  auf eine multimodale Verteilung abgebildet. Je dichter die Stichproben an einander liegen desto höher ist an dieser Stelle auch die Wahrscheinlichkeit.

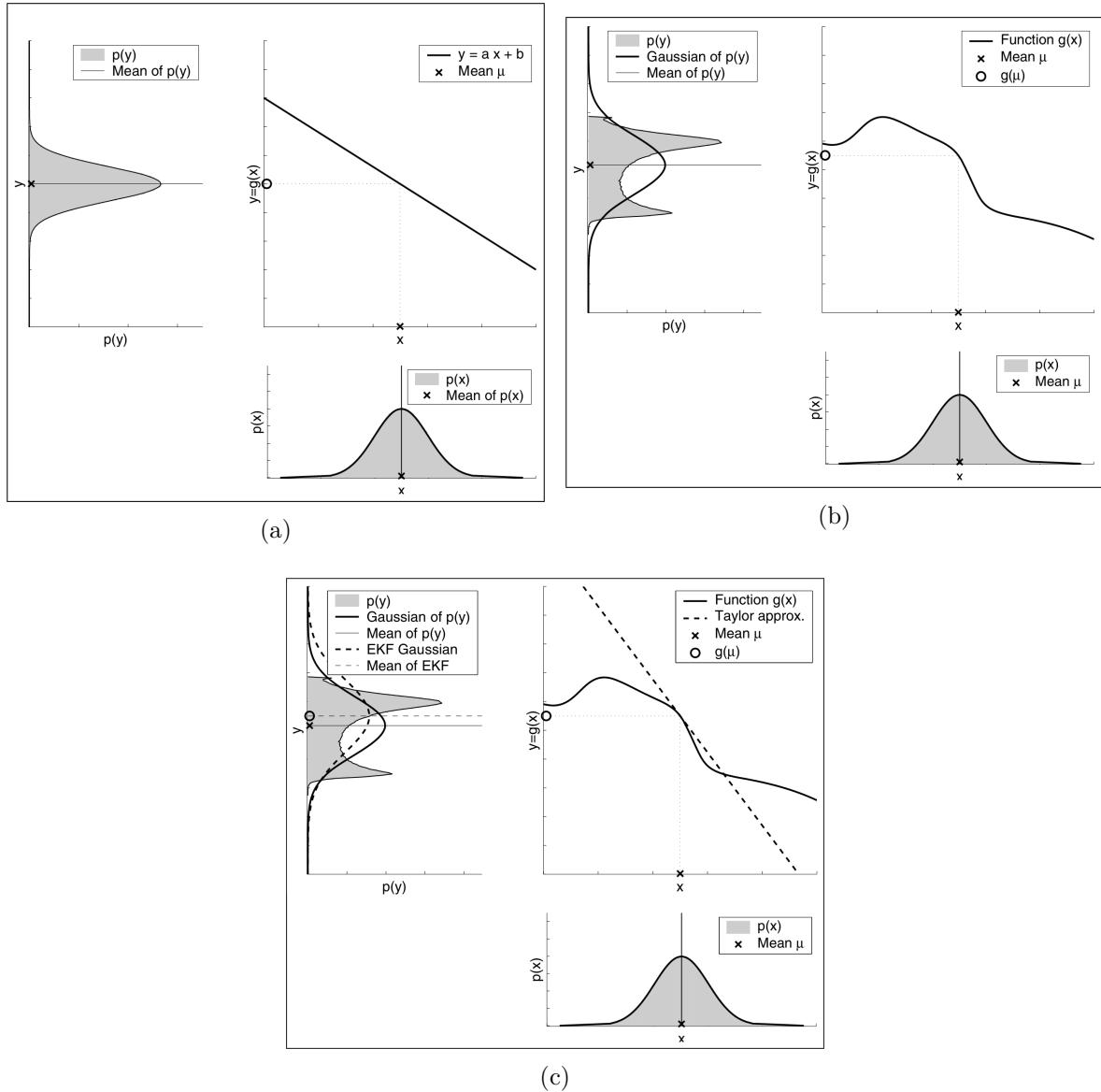


Abbildung 2.4.: Lineare und nicht lineare Transformation einer normalverteilten Zufallsvariable. [2]

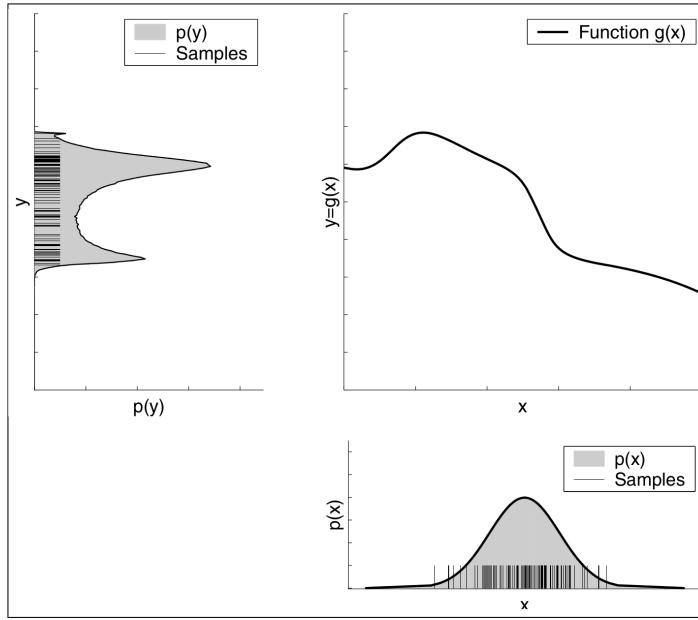


Abbildung 2.5.: Darstellung einer Zufallsvariable vor und nach der Abbildung, durch eine nicht lineare Funktion, mittels einem Partikel Filter. [2]

Die Stichproben aus der a posteriori Verteilung werden als Partikel (engl. Particle) bezeichnet. Jeder Partikel  $x_t$  entspricht dabei einer Hypothese des Weltzustandes zum Zeitpunkt  $t$ . Über ein Gewicht  $w_t$  wird die Relevanz (engl. Importance Weight) eines Partikels ausgedrückt. Die Idee ist es dabei, durch ein Set von Partikeln den Belief  $bel(x_t)$  möglichst gut anzunähern, siehe Gleichung 2.15. Der Parameter  $M$  entspricht dabei der Anzahl der Partikel und wird sehr großzügig gewählt, z.B.  $M = 1000$ .

$$\mathcal{X}_t := x_t^{[1]}, x_t^{[2]}, \dots, x_t^{[M]} \quad (2.15)$$

Bei dem Partikel Filter handelt es sich, wie bei dem Bayes Filter, ebenfalls um einen rekursiven Algorithmus, dem das vorherige Partikel-Set  $\mathcal{X}_{t-1}$ , die aktuellen Steuerbefehle  $u_t$  und Wahrnehmungen  $z_t$  übergeben werden, siehe Algorithmus 2.3. Dabei wird im ersten Schritt auf jedem Partikel, aus dem vorherigen Set, die aktuellen Steuerbefehle angewendet und somit eine neue Prognose für den aktuellen Zustand gebildet, siehe Zeile 4. Weiterhin wird mit der neuen Zustandsprognose und den aktuellen Wahrnehmungen das Gewicht des Partikels bestimmt, siehe Zeile 5. Die aktuelle Zustandsprognose sowie das Gewicht werden danach in dem temporären Set  $\bar{\mathcal{X}}_t$  gesammelt, siehe Zeile 6. Im zweiten Schritt findet das sogenannte Resampling oder Importance Sampling statt, siehe Zeile 9. Hierbei werden die Partikel mit einem hohen Gewicht beibehalten und die mit einem geringen durch neue ersetzt, die im Bereich der Partikel mit den hohen Gewichten liegen. Dadurch ist gewährleistet, dass sich die Partikel nicht in Bereichen mit einer geringen a posteriori Wahrscheinlichkeit aufhalten. Das Ergebnis aus dem letzten Schritt wird in dem Partikel-Set  $\mathcal{X}_t$  gespeichert und dem Aufrufer zurückgegeben.

```

1: Algorithm Particle_Filter( $\mathcal{X}_{t-1}$ ,  $u_t$ ,  $z_t$ ):
2:    $\bar{\mathcal{X}}_t = \mathcal{X} = \emptyset$ 
3:   for  $m = 1$  to  $M$  do
4:     sample  $x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$ 
5:      $w_t^{[m]} = p(z_t | x_t^{[m]})$ 
6:      $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$ 
7:   end
8:   for  $m = 1$  to  $M$  do
9:     draw  $i$  with probability  $\propto w_t^{[i]}$ 
10:    add  $x_t^{[i]}$  to  $\mathcal{X}_t$ 
11:   end
12: return  $\mathcal{X}_t$ 

```

Algorithmus 2.3 : Partikel Filter Algorithmus [2]

### 2.5.1. Monte Carlo Lokalisierung

Das Bestimmen der Pose eines Roboters relativ zu einer gegebenen Karte der Umwelt wird als Lokalisierung bezeichnet. Die Lokalisierung bestimmt dabei die Transformation zwischen dem lokalen Koordinatensystem des Roboters und dem globalen Koordinatensystem der Karte. Ist die initiale Pose des Roboters nicht gegeben, wird dieses Problem als globale Lokalisierung bezeichnet. Je größer die Karte der Umwelt ist, desto wahrscheinlicher wird es, dass die Wahrnehmung, Mehrdeutigkeiten in Bezug zu der Karte aufweist, die sich in einer multimodalen Verteilung widerspiegelt und damit ideal durch einen Partikel Filter abgebildet werden kann. Bei der Monte Carlo Lokalisierung (MCL) handelt es sich um genau so einen Partikel Filter, der mittlerweile zu den Standardalgorithmen der Robotik gehört.

Die zwingende Voraussetzung, für die Lokalisierung, ist das vorhanden sein einer sehr genauen Karte der Umwelt. Die Kartendaten  $m$  werden bei der Berechnung der Gewichte für die Zustandsprognosen eingesetzt. Somit unterscheidet sich der Monte Carlo Algorithmus nur in der fünften Zeile von dem generischen Partikel Filter Algorithmus, siehe Gleichung 2.16.

$$w_t^{[m]} = p(z_t | x_t^{[m]}, m) \quad (2.16)$$

## 2.6. Simultaneous Localization and Mapping

Wenn der Roboter weder seine eigene Pose kennt noch eine Karte seiner Umwelt besitzt, beides jedoch aus den Steuerbefehlen  $u_{1:t}$  und den Wahrnehmungen  $z_{1:t}$  bestimmen soll, wird das als SLAM<sup>1</sup> Problem bezeichnet.

Es gilt dabei zu unterscheiden zwischen dem Online- und Full-SLAM. Beim Online-SLAM wird die aktuelle Pose  $x_t$  innerhalb der Karte geschätzt, siehe Gleichung 2.17, während beim

---

<sup>1</sup>Alternativ zu Simultaneous Localization and Mapping (SLAM) wird auch Concurrent Mapping and Localization (CML) verwendet.

Full-SLAM die komplette Trajektorie  $x_{1:t}$  innerhalb der Karte geschätzt wird, siehe Gleichung 2.18.

$$p(x_t, m \mid z_{1:t}, u_{1:t}) \quad (2.17)$$

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t}) \quad (2.18)$$

### 2.6.1. EKF-SLAM

Eine der ersten Implementierung des Online-SLAM Algorithmus wurde auf der Basis des EKF durchgeführt. Neben der Pose des Roboters, werden auch alle Position der wahrgenommenen Landmarken geschätzt. Somit besteht die Karte der Umwelt aus allen Positionen der Landmarken und wird daher auch als merkmalsbasierte Karte (engl. Feature-Based Map) bezeichnet.

Bisher bestand der Zustandsvektor aus den drei Komponenten X-/Y-Position und der Orientierung  $\theta$  um die Pose des Roboters zu beschreiben. Für den EKF-SLAM wird der Zustandsvektor um die X-/Y-Position der vorhandenen Landmarken erweitert, siehe Gleichung 2.19. Diese Kombination aus Pose des Roboters und Positionen der Landmarken wird als kombinierter Zustandsvektor (engl. Combined State Vector) bezeichnet.

$$x_t = (x, y, \theta, m_{1,x}, m_{1,y}, \dots, m_{n,x}, m_{n,y})^T \quad (2.19)$$

Um den EKF einzusetzen zu können, muss der Belief mittels der beiden Momente  $\mu$  und  $\Sigma$  beschrieben werden.  $\mu$  entspricht dabei dem kombinierten Zustandsvektor  $x_t$  und  $\Sigma$  beschreibt die Unsicherheiten der Roboter Pose, der Landmarken Positionen und die Unsicherheit zwischen der Roboter Pose und den Landmarken Positionen, siehe Gleichung 2.20.

$$\mu = \begin{pmatrix} x_R \\ m_1 \\ \vdots \\ m_n \end{pmatrix} \quad \Sigma = \begin{pmatrix} \Sigma_{x_R x_R} & \Sigma_{x_R m_1} & \cdots & \Sigma_{x_R m_n} \\ \Sigma_{m_1 x_R} & \Sigma_{m_1 m_1} & \cdots & \Sigma_{m_1 m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{m_n x_R} & \Sigma_{m_n m_1} & \cdots & \Sigma_{m_n m_n} \end{pmatrix} \quad (2.20)$$

Bei der Verwendung des EKF-SLAM ist zu berücksichtigen, dass die Wahrnehmungsdaten möglichst eindeutig den Landmarken zugeordnet werden können. Die Zuordnung kann dabei explizit gegeben sein oder durch einen zusätzlichen Algorithmus möglich gut geschätzt werden. Mehrdeutige Zuordnungen führen zu einem deutlich schlechteren Ergebnis.

Eine weitere Einschränkung betrifft die maximale Anzahl der Landmarken, da die Laufzeit und die Größe der Karte quadratisch mit der Anzahl der Landmarken wächst.

### 2.6.2. FastSLAM

Bei dem FastSLAM Verfahren handelt es sich um eine Lösung für das SLAM Problem mithilfe eines Partikel Filters. Bei einem Partikel Filter repräsentiert jedes Partikel eine eigene Hypothese des jeweiligen Zustandes des Roboters. Um das SLAM Problem zu lösen muss

der Zustandsvektor, wie beim EKF-SLAM, um die Landmarken erweitert werden, siehe Gleichung 2.19.

Partikel Filter haben jedoch die Eigenschaft, nur mit einem kleinen Zustandsraum gut zu funktionieren. Bei einem großen Zustandsraum werden deutlich mehr Stichproben benötigt, um diesen möglichst gut abzudecken. Der Aufwand skaliert dabei exponentiell mit der Anzahl der Dimensionen des Zustandsraumes, sodass eine praktikable Anwendung eines Partikel Filter nicht mehr gegeben ist.

Die Lösung des Problems besteht darin, mit dem Partikel Filter nur den Pfad des Roboters zu modellieren und dann für jede Pfadhypothese eine separate Karte der Umwelt mit ihren Landmarken zu berechnen.

Um die Berechnung der Karte von dem Pfad des Roboters zu trennen, wird der Umstand ausgenutzt, das beide Zufallsvariablen ein Wissen über einander besitzt. Dazu wird die Gleichung 2.11, für die bedingte Wahrscheinlichkeit, nach der multivariaten Verteilung umgestellt, siehe Gleichung 2.21. Anstatt nun die Verteilung sowohl für den Pfad des Roboters als auch den der Karte  $p(x, y)$  zu bestimmen, wird die Verteilung nur für den Pfad des Roboters  $p(x)$  bestimmt und dann für jede Pfadhypothese die sich daraus ergebende Karte  $p(y | x)$  berechnet. Dieses Verfahren wird auch als Rao-Blackwellisierung bezeichnet.

$$p(x, y) = p(y | x) p(x) \quad (2.21)$$

Das vorherige Verfahren wird nun auf die a posteriori Verteilung des SLAM Problems übertragen, siehe Gleichung 2.22.  $x_{0:t}$  entspricht dabei den Pfad des Roboters,  $m_{1:M}$  der Karte,  $z_{1:t}$  den Wahrnehmungen und  $u_{1:t}$  den Steuerbefehlen. Die ausfaktorisierten Terme  $p(x_{0:t} | z_{1:t}, u_{1:t})$  und  $p(m_{1:M} | x_{0:t}, z_{1:t})$  bilden dabei die a posteriori Verteilung für den Pfad des Roboters und für die Karte der Umwelt.

$$p(x_{0:t}, m_{1:M} | z_{1:t}, u_{1:t}) = p(x_{0:t} | z_{1:t}, u_{1:t}) p(m_{1:M} | x_{0:t}, z_{1:t}) \quad (2.22)$$

Der Term für die a posteriori Verteilung der Karte aus der Gleichung 2.22 kann noch optimiert werden. Dadurch das der Pfad des Roboters bekannt ist, gibt es keine Landmarke die über eine unbekannte Zufallsvariable mit einer anderen Landmarke verbunden ist, siehe Abbildung 2.6. Das bedeutet, dass die Landmarken untereinander als unabhängige Zufallsvariable dargestellt werden können. Somit kann die a posteriori Verteilung der Karte als Produkt ihrer einzelnen Landmarken ausgedrückt werden, siehe Gleichung 2.23. Dadurch kann jede Landmarke durch einen  $2 \times 2$  großen EKF modelliert und dadurch sehr effizient berechnet werden. Die Repräsentation des endgültigen Partikel kann der Abbildung 2.7 entnommen werden.

$$p(x_{0:t}, m_{1:M} | z_{1:t}, u_{1:t}) = p(x_{0:t} | z_{1:t}, u_{1:t}) \prod_{i=1}^M p(m_i | x_{0:t}, z_{1:t}) \quad (2.23)$$

Die Gewichte eines Partikels spielen eine große Rolle im Resampling Prozess. Beim FastSLAM entsteht die Gewichtung dabei aus der Erkenntnis, wie gut die Landmarken einer Pfadhypothese mit den Wahrnehmungen übereinstimmen.

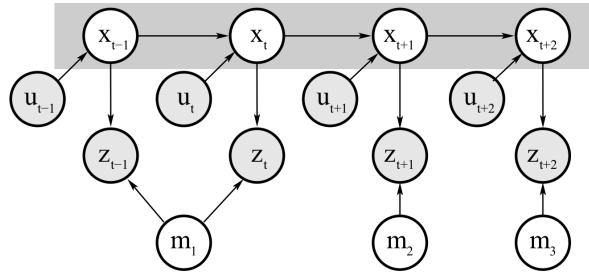


Abbildung 2.6.: Darstellung des SLAM Problems als bayessches Netz. [2]

	robot path	feature 1	feature 2	...	feature $N$
Particle $k = 1$	$x_{1:t}^{[1]} = \{(x \ y \ \theta)^T\}_{1:t}^{[1]}$	$\mu_1^{[1]}, \Sigma_1^{[1]}$	$\mu_2^{[1]}, \Sigma_2^{[1]}$	...	$\mu_N^{[1]}, \Sigma_N^{[1]}$
Particle $k = 2$	$x_{1:t}^{[2]} = \{(x \ y \ \theta)^T\}_{1:t}^{[2]}$	$\mu_1^{[2]}, \Sigma_1^{[2]}$	$\mu_2^{[2]}, \Sigma_2^{[2]}$	...	$\mu_N^{[2]}, \Sigma_N^{[2]}$
⋮					
Particle $k = M$	$x_{1:t}^{[M]} = \{(x \ y \ \theta)^T\}_{1:t}^{[M]}$	$\mu_1^{[M]}, \Sigma_1^{[M]}$	$\mu_2^{[M]}, \Sigma_2^{[M]}$	...	$\mu_N^{[M]}, \Sigma_N^{[M]}$

Abbildung 2.7.: Representation der Partikel in FastSLAM. [2]

Für den EKF-SLAM ist es entscheidend, die Zuordnung zwischen den Landmarken und den Wahrnehmungen zu kennen. Das ist beim FastSLAM nicht mehr notwendig, da jede Pfadhypothese über eine eigene Zuordnung verfügt, siehe Abbildung 2.8. In Kombination mit den Gewichten pro Partikel, werden die Partikel mit der richtigen Zuordnung eher im Resampling Prozess ausgewählt als die mit einer falschen Zuordnung.

Der FastSLAM Algorithmus bildet die Grundlage für die beiden RO-SLAM Verfahren die zum einen in den Abschnitten 3.3 und 3.4 vorgestellt und zum anderen im der Evaluation ausgewertet werden.

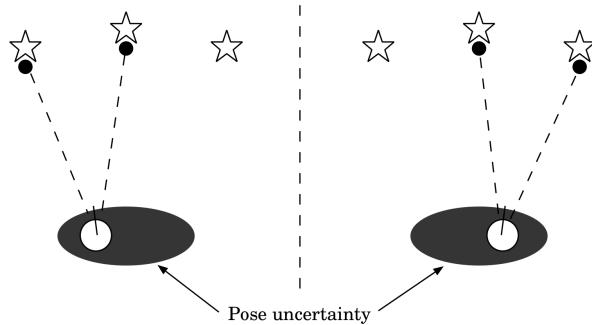


Abbildung 2.8.: Zuordnung der Landmarken zu den Wahrnehmungen pro Pfadhypothese. [2]

## 2.7. Robot Operating System

Bei dem ROS handelt es sich nicht im eigentlichen Sinne um ein Betriebssystem, sondern vielmehr um ein Framework das die Kommunikation zwischen verschiedenen Verarbeitungseinheiten regelt. Im Jahre 2007 begann die Entwicklung von ROS an der Stanford University. Ab 2009 wurde dieses dann hauptsächlich an dem Robotik Institut Willow Garage weiterentwickelt. Durch die BSD Lizenz steht ROS, als Open-Source-Projekt, sowohl der nicht-kommerziellen als auch der kommerziellen Weiterentwicklung zur Verfügung. [9]

Jedes ROS System muss über einen ROS Master verfügen. Dieser stellt den zentralen Punkt für die Registrierung von Knoten (engl. Nodes), Datenbusse (engl. Topics) und Dienste (engl. Services) zur Verfügung.

Jeder Verarbeitungseinheit in ROS wird durch einen Knoten repräsentiert. Ziel ist es möglichst kleine, wiederverwendbare und miteinander kombinierbare Einheiten zu bilden. Die Programmierung eines Knoten erfolgt dabei in den Programmiersprachen C++, Python oder Lisp.

Die Kommunikation zwischen den Knoten erfolgt dabei über ein Peer-to-Peer (P2P) Kanal auf der Basis von Nachrichten (engl. Messages). Bei einer Nachricht handelt es sich um eine Datenstruktur die primitive Datentypen, die Datentypstruktur anderer Nachrichten und Felder enthalten kann. Eine Nachricht wird dabei über das Publish-Subscribe Muster in einem Datenbus veröffentlicht und kann von jedem Knoten empfangen werden der Nachrichten von diesem Datenbus abonniert hat. Der typische Anwendungsfall für dieses Verfahren ist die Bereitstellung von Sensordaten und Rückmeldungen über Statusänderungen.

Das zuvor vorgestellte Publish-Subscribe Muster stellt eine asynchrone Kommunikation bereit. Mit Hilfe von Diensten ist auch ein synchroner Nachrichtenaustausch möglich. Hierfür wird eine Anfrage von dem Client an den Server gestellt, der seinerseits mit einem Ergebnis antwortet. Dieses Verfahren entspricht dem Request-Response Muster. Die Erstellung einer inversen Transformation ist ein typischer Anwendungsfall hierfür.

Nach der Entwicklung neuer Algorithmen ist ein Vergleich mit den bereits bestehenden Algorithmen von großem Interesse. Mit dem Konzept der Bag-Dateien können alle veröffentlichten Nachrichten aufgezeichnet werden und zu einem späteren Zeitpunkt wieder abgespielt werden.

Um die Wiederverwendbarkeit von Knoten, Nachrichten und Diensten zu fördern, wurde das Konzept der Pakete (engl. Packages) eingeführt. Ein Paket stellt die kleinste erstellbare Einheit dar und beinhaltet alle Teile eines Softwarepaketes wie z.B. Quellcode-, Konfigurationsdateien, Drittanbieter-Bibliothek, Abhängigkeitslisten usw.

Eine Roboterplattform besteht aus vielen Sensoren und Aktuatoren die abgefragt und gesteuert werden müssen. Dementsprechend viele Knoten müssen gestartet werden. Diese Aufgabe erfüllen die Startdateien (engl. Launch-Files). Neben der Definition der Knoten die gestartet werden sollen, können Einzelne oder eine Gruppe von Knoten parametrisiert werden. Mittels der Verschachtelung von Startdateien ist eine Wiederverwendung von knotenspezifischen Startdateien möglich.

Zu den häufigsten Operationen der Verarbeitungseinheit einer Roboterplattform ist das Transformieren der Sensordaten aus dem Sensorkoordinatensystem in das Koordinatensystem

des Robotermittelpunktes. In ROS wird hier für ein Transformationsbaum (engl. TF-Tree) verwendet, der aus statischen und dynamischen Transformationen besteht. Statische Transformationen werden dort eingesetzt, wo sich die Pose zweier Koordinatensysteme zur Laufzeit nicht ändert, z.B. zwischen dem Robotermittelpunkt und einem fest montierten Sensor. Dynamische hingegen bei Koordinatensystemen die sich zur Laufzeit ändern, z.B. zwischen dem Robotermittelpunkt und den Inkrementalgebern der Antriebseinheit. Statische Transformationen können mittels einer Unified Robot Description Format (URDF) Datei modelliert und allen Knoten bereitgestellt werden.

# 3.

## Stand der Forschung und Technik

In diesem Kapitel wird ein chronologischer Überblick über die wichtigsten Publikationen zum Thema RO-SLAM geliefert. Jeder Autor verwendet eine etwas andere Terminologie, auch wenn es sich im Kern um die gleiche Sache handelt. Daher werden in diesem Kapitel die Begriffe Anker, Tag und Beacon synonym verwendet.

### 3.1. Preliminary results in range-only localization and mapping

In „Preliminary results in range-only localization and mapping“ [10] verwendet der Author Beacons die eine Entfernungsschätzung mit einem mittleren Fehler von 1,82 m (6 Fuß) liefern. Um das Problem der Datenzuordnung zu trivialisieren, liefert jeder Beacon zusätzlich zur Messung eine eindeutigen Identifikationsnummer mit. Nur mit diesen Daten werden die Probleme der Roboter Lokalisierung, Positionsverfolgung und SLAM behandelt. Das algorithmische Fundament wird dabei durch den EKF und Monte Carlo (MC) Partikel Filter gebildet.

Im ersten Abschnitt wird die statische Lokalisierung thematisiert, die für eine Positions schätzung auf vorherige Sensorinformationen und Positionsschätzungen verzichtet. Bei einer fehlerfreien Entfernungsmessungen ist die Positionsbestimmung ein mathematisch triviales Problem. Da die Messungenauigkeit mit 1,82 m jedoch sehr hoch ist, werden Methoden der Wahrscheinlichkeitstheorie verwendet. Im ersten Schritt wird die Verteilungsfunktion experimentell für einige diskrete Abstände bestimmt. Unter zuhilfenahme der erstellten Verteilungsfunktion wird für jeden Beacon und für jede Zelle des Probability Grid die Wahrscheinlichkeit berechnet, siehe Abbildung 3.1a links. Im letzten Schritt werden die einzelnen Probability Grid miteinander multipliziert und im Abschluss skaliert damit die Summe über alle Zellen gleich eins ist, siehe Abbildung 3.1b rechts. Aus dem gewichteten Durchschnitt der Zellen lässt sich die Position schätzen, siehe Abbildung 3.1b. Der durchschnittliche Schätzungsfehler beträgt 0,49 m bei einem erwarteten Fehler von 1,77 m–2,18 m bei der Entfernungsmessung.

Im nächsten Abschnitt wird die Positionsverfolgung mittels eines EKF bzw. MC Partikel Filter behandelt. Hierfür werden die Odometriedaten und die vorherige Positionsschätzung verwendet. Für eine initiale Positionsschätzung wird das Verfahren aus dem vorherigen Abschnitt verwendet. Aus der Entfernungsschätzung eines Beacons ergibt sich eine ringförmige Verteilungsfunktion, die mit einer unimodalen Verteilung nicht modelliert werden kann. Als Annäherung wird eine Normalverteilung verwendet, die in tangentialer Richtung gestreckt

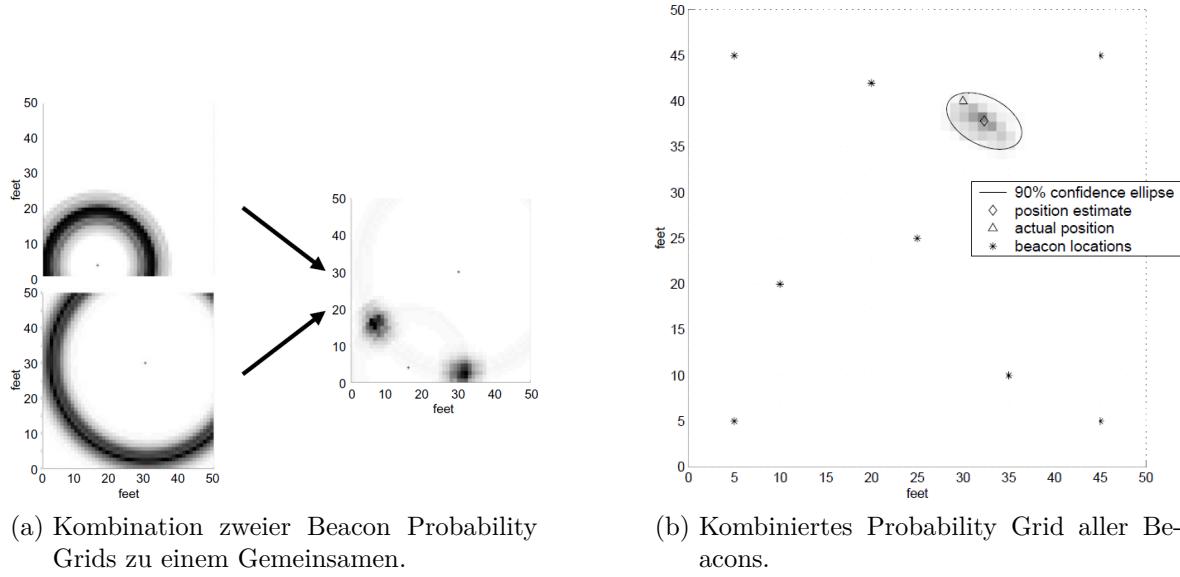


Abbildung 3.1.: Darstellung der Beacons in einem Probability Grid. [10]

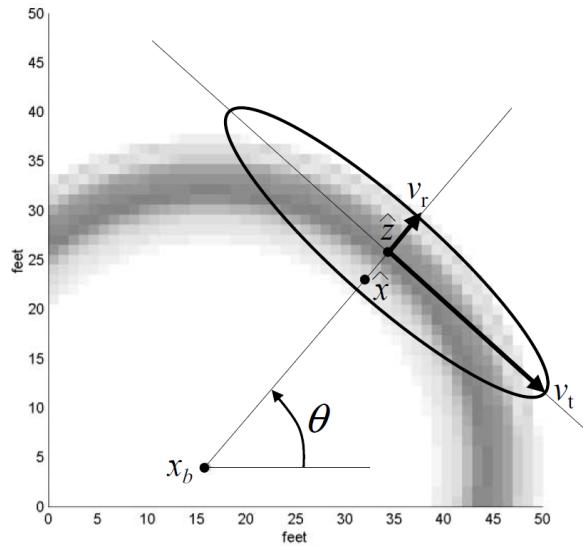


Abbildung 3.2.: Annäherung einer ringförmigen Verteilung an eine Normalverteilung. [10]

und in radialer Richtung gestaucht ist. Als Mittelwert wird die vorherige Positionsschätzung verwendet, siehe Abbildung 3.2. Um die Partikel im MC Partikel Filter zu gewichten ist die ringförmige Verteilung vollkommen ausreichend. Der durchschnittliche Schätzungsfehler beträgt beim EKF 0,22 m und 0,28 m beim MC Partikel Filter.

Im letzten Abschnitt wird ein EKF-SLAM Verfahren eingesetzt. Die initiale Roboter- und Beacon-Position muss nicht genau jedoch ungefähr bekannt sein. Der Zustandsvektor des EKF beinhaltet die Roboter- und alle Beacon-Positionen. Der durchschnittliche Fehler in der initialen Schätzung beträgt 1,56 m und verbessert sich zum Schluss hin auf 0,23 m.

### 3.2. Robust range-only beacon localization

In „Robust range-only beacon localization“ [11] werden autonome Unterwasserfahrzeuge (engl. Autonomous Underwater Vehicle (AUV)) verwendet um den Ozeanen der Welt die letzten Geheimnisse zu entlocken. Hierfür ist eine genaue Lokalisierung der AUV notwendig. Dies wird über die Messung der Signallaufzeit zwischen dem AUV und mehreren stationären akustischen Transponder-Beacon erreicht. Vor der eigentlichen Messung müssen jedoch die Beacons im Messbereich verteilt werden, deren Position genau bestimmt werden und zusätzlich dafür gesorgt werden, dass die Beacons ihre Position nicht ändern. Dadurch handelt es sich um ein sehr kostspieliges Prozedere und soll in dem vorgestellten Verfahren dahingehend optimierte werden, dass die Beacon-Position vorher nicht bekannt sein muss. Dadurch ist eine autonome Verteilung der Beacons möglich und zusätzlich kann auch detektiert werden ob ein Beacon seine Position verändert hat.

Bevor jedoch die Laufzeitmessungen verwendet werden können, müssen diese um Ausreißer (engl. Outlier) bereinigt werden. Diese entstehen z.B. durch unterschiedliche Ausbreitungsgeschwindigkeiten der Schallwellen im Wasser, durch Reflexionen am Meeresgrund (engl. Multipath) oder durch Interferenzen der Nutzlast (engl. Payload) mit den Sensoren. Die Bereinigung der Messdaten erfolgt hierbei über das Spectral Graph Partitioning Verfahren. Hierbei wird jede Messung als Kreis dargestellt. Überschneiden sich zwei Kreise gelten diese beiden Messungen als konsistent. Jede der Messungen wird in einem Graphen zu einem Vertex und jede konsistente Messung zu einer Kante. Bei dem fertigen Graphen ist nun zu beobachten, dass Outlier weniger stark untereinander verbunden sind als Inlier. Diese durchschnittliche Konnektivität der Inlier wird als Metrik für den Partitionierungsalgorithmus verwendet um die Messdaten zu filtern.

Nachdem die Messwerte pro Beacon gefiltert worden sind, werden zwei Messwerte von zwei verschiedenen Beacons wieder mit einander überschnitten. Hieraus entstehen zwei mögliche Positionen des AUV. In einem Grid erhöhen diese beiden Positionen dann den Ihnen zugeordneten Zähler. Dieser Vorgang wird für weitere Messwerte wiederholt. Durch das daraus resultierende Voting-System entstehen zwei Gipfel (engl. Peak) die die mögliche Position des AUV approximieren. Ist der Höhenunterschied zwischen den Gipfeln ausreichend groß, wird die AUV-Position mit dem höchsten Gipfel an einen EKF-SLAM übergeben.

### 3.3. A pure probabilistic approach to range-only SLAM

Alle zuvor vorgestellten Verfahren hatten den Nachteil, dass sie probabilistische und nicht probabilistische Verfahren gemeinsam verwendeten um eine initiale Schätzung der Beacon-Positionen zu erlangen. In dieser Initialisierungsphase waren alle Positionsinformationen für die Positionsschätzung des Roboters verloren. In der Arbeit „A pure probabilistic approach to range-only SLAM“ [12] wird zu jedem Zeitpunkt die beste Beacon-Schätzung genutzt um die Roboterposition zu verbessern. Um das zu bewerkstelligen, wird ein rein probabilistisches Verfahren genutzt.

Verwendung findet hierbei der Rao-Blackwellized Particle Filter (RBPF), der die Berechnung des Roboterpfades von der Karte mit den Landmarken entkoppelt [13, 14]. Jeder Partikel

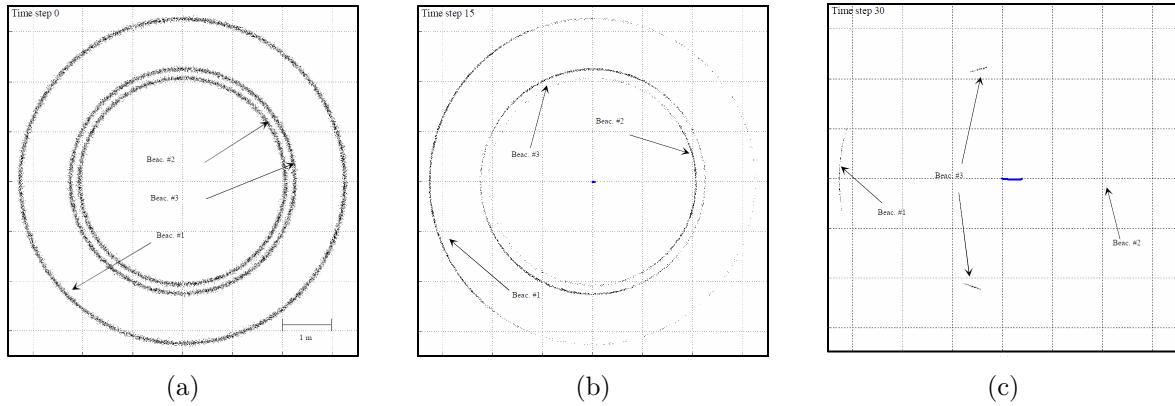


Abbildung 3.3.: Entwicklung eines Hilfspartikel Filters zu einem EKF. [12]

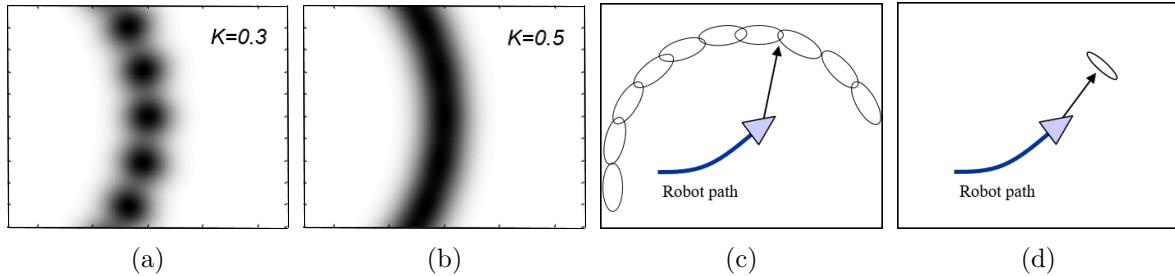


Abbildung 3.4.: Modellierung der ringförmigen Verteilung über radial angeordnete Normalverteilungen. [16]

des RBPF beschreibt dabei pro Beacon die Verteilung die am besten zu seiner Positionsschätzung passt. Die Verteilung wird dabei zuerst von einem Hilfspartikel Filter (engl. Auxiliary Particle Filter) modelliert. Seine Partikel besitzen dabei eine ringförmige Verteilung mit dem Abstand des Beacon, siehe Abbildung 3.3a. Partikel mit einer geringen Wahrscheinlichkeit werden über eine Gewichtung aussortiert, siehe Abbildung 3.3b. Sobald der Hilfspartikel Filter gegen eine bestimmte Beacon-Position konvergiert ist, wird dieser in einen Normalverteilung umgewandelt und über einen EKF modelliert, siehe Abbildung 3.3c.

### 3.4. Efficient probabilistic range-only SLAM

Aufbauend auf der vorherigen Arbeit im Abschnitt 3.3 wird in „Efficient probabilistic range-only SLAM“ [15] nicht mehr ein Hilfspartikel Filter verwendet, der ab einer gewissen Sicherheit in einen EKF umgewandelt wird. Stattdessen wird die ringförmige Verteilung über radial angeordnete Normalverteilungen angenähert deren Nachbarschaftsabstand über den Parameter  $K$  bestimmt werden kann, siehe Abbildung 3.4a – 3.4b. Die Gesamtverteilung ergibt sich dabei über eine gewichtete Summe der Normalverteilungen (engl. Sum of Gaussians (SOG)). Über die Gewichte lässt sich dabei steuern, ob eine der Normalverteilungen in dem Ring für die Beacon-Position noch relevant ist oder entfernt werden kann, siehe Abbildung 3.4c – 3.4d.

Sowohl das Verfahren aus dem Abschnitt 3.3 als auch dieses sind in dem MRPT Framework vorhanden und werden im Kapitel 5 verwendet.

### 3.5. A robust method of localization and mapping using only range

Jede bisherige Veröffentlichung die einen RO-SLAM mittels eines EKF umsetzen will, muss mit dem Problem der Linearisierung und der Multimodalität der Verteilung kämpfen. Hierfür wurde das Problem bisher immer im kartesischen Koordinatensystem betrachtet. Die Autoren dieses Werkes [17] sind einen anderen Weg gegangen und haben das Problem ins Polarkoordinatensystem überführt. Dieses Vorgehen hat den großen Vorteil, dass ringförmige Verteilungen sehr gut im Polarkoordinatensystem modelliert werden können. Auch ist eine Vorverarbeitung (engl. Batch Process) der Entfernungsmessungen nicht mehr notwendig. Die Ergebnisse der Messung kann direkt an den Filter übergeben werden.

Wie bereits im Grundlagenkapitel besprochen besteht der Zustandsvektor eines EKF für die Lokalisierung aus der Pose (Position und Orientierung) des Roboters:

$$q_k = [x_k^r, y_k^r, \phi_k^r]^T.$$

Für das SLAM Problem muss der Zustandsvektor um die Positionen der Tags erweitert werden:

$$q_k = [x_k^r, y_k^r, \phi_k^r, x_k^1, y_k^1, \dots, x_k^N, y_k^N]^T.$$

Die Beschreibung des Zustandsvektors in Polarkoordinaten erfolgt dabei über die Festlegung der Mitte des Polarkoordinatensystems ( $c_{x,k}, c_{y,k}$ ) und über die Entfernung und den Winkel ( $r_k, \theta_k$ ):

$$q_k = [c_{x,k}^r, c_{y,k}^r, r_k^r, \theta_k^r, c_{x,k}^1, c_{y,k}^1, r_k^1, \theta_k^1, \dots, c_{x,k}^N, c_{y,k}^N, r_k^N, \theta_k^N]^T. \text{ } ^1$$

Damit können alle Tags beschrieben werden, für den Roboter benötigt man aber noch den zusätzlichen Parameter  $\phi_k^r$  um die Ausrichtung (engl. Heading) festzulegen.

Das vorherige Bewegungsmodell (engl. Motion Model) für das kartesischen Koordinatensystem kann weiterverwendet werden, nur die Parameterzuweisungen an  $(c_{x,k}^r, c_{y,k}^r, \phi_k^r)$  ändern sich. Einzig das Messmodell (engl. Measurement Model) muss angepasst werden, um die erwartete Entfernung zu berechnen. Hierfür findet eine Konvertierung der Polarkoordinaten in Kartesische Koordinaten statt.

Die Abbildungen 3.5 verdeutlichen zum einen, wie die Verteilung in Polarkoordinaten dargestellt wird und zum andern, wie mehrere Messungen miteinander kombiniert werden.

Abbildung 3.5a beschreibt die tatsächliche Verteilung einer Einzelmessung. Das blaue Rechteck repräsentiert die Roboterposition und der rote Diamant die tatsächliche Tagposition. Die tatsächliche Verteilung einer Einzelmessung wird in Polarkoordinaten als grüne Fläche modelliert, siehe Abbildung 3.5b. Über die blaue Ellipse wird die tatsächliche Verteilung mittels einer Normalverteilung angenähert und ergibt dabei die ringförmige Verteilung in der Abbildung 3.5c.

Nachdem zwei Messungen von verschiedenen Positionen aus durchgeführt wurden, ergibt sich die tatsächliche Verteilung aus den beiden Schnittpunkten der Ringe, siehe Abbildung 3.5d.

---

<sup>1</sup>Da für die Beschreibung der Zustände deutlich mehr Parameter benötigt werden, bezeichnet man dieses auch als relative Überparameterisierung (engl. Relative-Over Parameterization (ROP)).

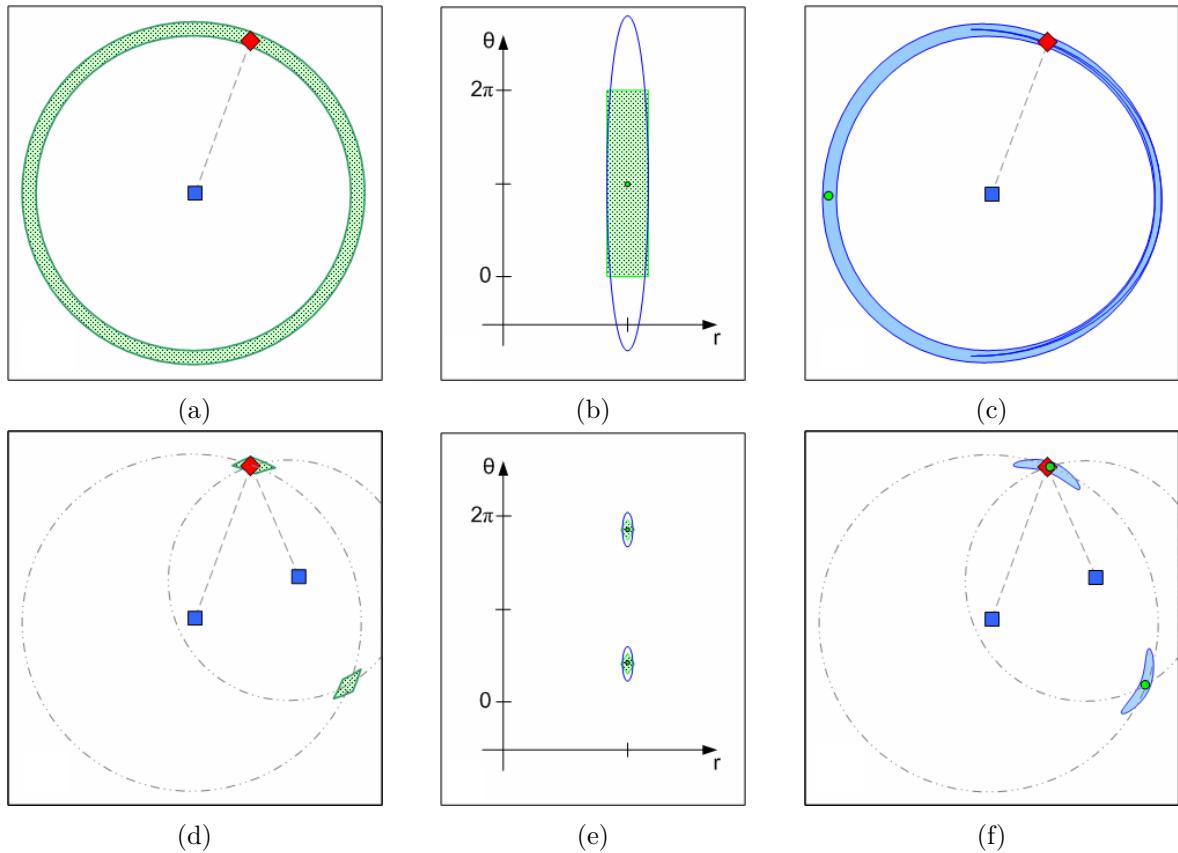


Abbildung 3.5.: Darstellung der tatsächlichen und angenäherten Verteilung in Kartesischen und Polarkoordinaten. [17]

Die Multimodalität der Verteilung kann über einen EKF nicht mehr abgebildet werden, daher wird zu diesem Zeitpunkt pro Verteilung ein eigener EKF angelegt. Abbildung 3.5e beschreibt wie im vorherigen Abschnitt die tatsächlichen und angenäherten Verteilungen in Polarkoordinaten und Abbildung 3.5f die Verteilung die sich aus der Annäherung der Verteilung ergibt.

# 4.

---

## Ultra-Wideband

---

Bei der normalen Funkübertragung werden die zu übermittelnden Basisbandsignale auf eine sinusförmige Trägerfrequenz auf moduliert. Bei der UWB Übertragung ist das nicht der Fall, die Signale werden im Basisband übertragen. Hierfür werden im Nanosekundenbereich kurze Impulse erzeugt. UWB Signale verwenden eine Bandbreite von bis zu 500 MHz und können im Frequenzbereich von 3,6 GHz–10,1 GHz eingesetzt werden. Um zu verhindern das bestehende Dienste wie GPS oder Wireless Local Area Network (WLAN) gestört werden, wurde durch die Federal Communications Commission (FCC) die maximale Abgabeleistung von UWB-Sendern stark begrenzt (unter  $-41,3 \text{ dBm/MHz}$ ). Dadurch ist eine Koexistenz mit den bereits bestehenden Diensten möglich, ohne dass diese sich gegenseitig stören. [18–22]

Zu den bemerkenswerten Eigenschaften der UWB Technologie zählt das verlustfreie Durchdringen von Hindernissen, präzise Entfernungsmessung im Zentimeterbereich, eine hohe Datenübertragungsrate von über 110 MB/s in einem Bereich von 10 m–15 m und der geringe Energieverbrauch. [19]

Die Einsatzgebiete werden dabei in die Klassen Bildgebende-, Kommunikation-, Mess- und Fahrzeugradarsysteme unterteilt. Der Bereich der Bildgebendensysteme ist weiter unterteilt in Bodenradar, Medizinische Überwachung und Diagnostik und in Überwachungssysteme die Objekte durch Wände erkennen können. Kommunikations- und Messsysteme werden ihrerseits unterteilt in Indoor- und Outdoor-Systeme unterteilt. Hierzu zählen auch Systeme für die Navigation, Güternachverfolgung und Hochgeschwindigkeitsdatenübertragung. Die letzte Klasse wird in die Bereiche der Kollisionsvermeidung und Abstandssysteme unterteilt. [19, 23, 24]

### 4.1. Historie

Als Vater der UWB Kommunikation kann der italienische Funkpionier Guglielmo Marconi angesehen werden. In den späten 1890er Jahren entwickelte er den Knallfunkensender, der über eine Funkenstrecke ein hochfrequentes Signal zur Übertragung von Morsezeichen erzeugt. Mit dieser Apparatur gelang es Ihm, im Jahre 1901 einen Nachrichtenaustausch zwischen Nordamerika und Europa über den Nordatlantik durchzuführen. [20]

Bis in die Anfänge der 1960er Jahre dominierte jedoch die sinusförmige Funkübertragungsform. Dies änderte sich als die Forscher vom Lawrence Livermore National Laboratory (LLNL) und Los Alamos National Laboratory (LANL) begangen die Ausbreitung elektromagne-

tischer Wellen nicht nur im Frequenz- sondern auch im Zeitbereich zu untersuchen. Grundlegende Erkenntnisse wurden dabei im Bereich der Impulssender, -empfänger und -antennen gesammelt. [20, 21, 23, 25]

Durch die Einführung der zeitbereichsbasierten Abtastoszilloskope im Jahre 1962 durch Tektronix bzw. Hewlett-Packard war es zum ersten Mal möglich eine UWB Wellenform aufzufangen und anzuzeigen. Ermöglicht wurde dies erst durch den Einsatz von Tunneldioden und Avalanchetransistoren. [20, 21, 23]

Ab dem Jahre 1964 produzierten beide Hersteller Messgeräte für die Diagnose im Zeitbereich. [26]

Ab den Anfängen der 1970er Jahre waren alle wichtigen Grundsteine für ein UWB System für Kommunikation- bzw. Radaranwendungen gelegt. Dazu zählten auch diverse eingereichte Patente von Harmuth an der Catholic University of America (CUA), Ross und Robbins bei der Sperry Rand Corporation und Paul van Etten an der United States Air Force (USAF) im Rome Air Development Center. [19, 20, 26] Hervorzuheben ist das eingereichte Patent von Ross im Jahre 1973, siehe [27].

Kurz darauf im Jahre 1974 wurde die UWB Technologie kommerziell erfolgreich von Morey bei der Geophysical Survey Systems Inc. (GSSI) für ein Bodenradar (engl. Ground Penetrating Radar (GPR)) angewendet. [26]

Im Zeitraum von 1977 bis 1989 wurden mehrere Programme und Workshops organisiert um die Entwicklung von UWB Systemen voranzutreiben, darunter auch bei der USAF und dem United States Department of Defense (USDOD). Ebenfalls gab es mehrere akademische Programme an diversen Instituten, darunter auch am LLNL, LANL, University of Michigan, University of Rochester und Polytechnic University, mit dem Fokus auf den physikalischen Unterschieden zwischen der Kurzimpulsübertragung und den Langimpulssignalen bzw. kontinuierlichen Impulssignalen bei der Interaktion mit verschiedenen Materialien. [26]

Ab dem Jahre 1989 wurde der Name UWB<sup>1</sup> durch das USDOD geprägt. Diese Definition galt für alle Geräte die mindestens eine Bandbreite von 1,5 GHz bzw. 25 % der Fractional Bandwidth (FBW) belegten. [19–21, 25, 28]

Im Jahre 1994 wurde von McEwan an der LLNL das Micropower Impulse Radar (MIR) konstruiert. Hierbei handelte es sich um ein UWB Radarsystem mit bemerkenswerten Eigenschaften. Das Radarsystem verfügte über eine sehr hohe Signalsensitivität, einen kompakten Aufbau, eine kostengünstige Herstellung und einen sehr geringen Energieverbrauch, der sich im Bereich von Mikrowatt befanden und daher ideal für batteriebetriebene Anwendung eignete. [26]

Vor dem Jahre 2002 war die Verwendung von UWB auf Radarssystem beschränkt, die größtenteils in militärischen Anwendungen aufzufinden waren. [19] Das änderte sich ab dem Jahre 1998, als die FCC mit der Standardisierung der UWB Nutzung begann. Im Jahre 2002 wurden durch die FCC in den Vereinigten Staaten von Amerika große Frequenzbereiche (3,6 GHz–10,1 GHz) für die kommerzielle Nutzung freigegeben hat, siehe First Report and

<sup>1</sup>Vorher war die UWB Technologie nur unter den Synonymen „baseband communication“, „carrier free communication“, „impulse radio“, „large relative bandwidth communication“, „nonsinusoidal communication“, „orthogonal functions“, „sequency theory“, „time domain“, „large-relative-bandwidth radio/radar signals“, „video-pulse transmission“ oder „Walsh waves communication“ bekannt.

Order (R&O). Danach wurden erstmals auch eine nicht militärische Anwendung im Bereich „Imaging Systems“, „Communication and Measurement Systems“ und „Vehicular Radar Systems“ möglich. [19]

Weitere Staaten folgten der FCC Regulierung/Standardisierung und gaben ebenfalls große Frequenzbereiche für die UWB Technologie frei. Details zu den Regularien der einzelnen Staaten können unter [29] eingesehen werden.

## 4.2. Erstellte Hardware

### 4.2.1. Anforderungen

An die zu erstellende Hardware werden mehrere Anforderungen gestellt.

Um eine Entfernungsmessung durchzuführen wird immer ein Tag und mindestens ein Anker benötigt. Sowohl der Tag als auch der Anker sollen aus den gleichen elektrischen Komponenten bestehen, also eine gemeinsame Hardwareplattform bilden. Die unterschiedliche Funktionalität pro Modul soll sich dann aus verschiedenen Software-Ständen der Firmware herausbilden.

Die Anker sollen im Bedarfsfall frei im Raum verteilt werden können. Nicht an jeder Stelle steht eine Stromversorgung zur Verfügung, daher muss jedes Modul über eine separate Energiequelle verfügen.

Zusätzlich muss der Tag über eine bidirektionale Kommunikationsschnittstelle zur Verarbeitungseinheit verfügen. Über diese sollen zum einen Steuerbefehle an das UWB Modul geschickt werden und zum anderen sollen die gemessenen Entfernungen zwischen dem Tag und den Ankern an die Verarbeitungseinheit übertragen werden.

### 4.2.2. Hardware Zusammenstellung

#### UWB Transceiver

Als UWB Transceiver werden die Komponenten der Firma DecaWave verwendet. Bei dem DW1000 handelt es sich nur um dem Integrated Circuit (IC) der für das Erzeugen und Verarbeiten der UWB Funksignale zuständig ist. Der DWM1000 beinhaltet neben dem DW1000 auch die notwendige Beschaltung und zusätzlich eine Antenne für die Übertragung, siehe Abbildung 4.1.

Der DWM1000 kann mit einer Spannung von 2,8 V–3,6 V [31] betrieben werden, idealerweise mit 3,3 V. Das bedeutet aber auch, dass die Logikpegelspannung für die Serial Peripheral Interface (SPI) Schnittstelle 3,3 V beträgt. Dieser Umstand muss bei der Auswahl des Mikrocontrollers berücksichtigt werden.

Die Kommunikation mit dem DWM1000 erfolgt über die SPI Schnittstelle, hierfür sind die Pins Serial Clock (SCLK), Master Output Slave Input (MOSI), Master Input Slave Output (MISO) und Slave Select (SS) zu verwenden. [31] Bei der SPI-Schnittstelle handelt es sich um eine Master-Slave Architektur, das bedeutet das Daten vom Master gesendet und angefragt werden können. Der Slave kann jedoch nur Daten auf Anfrage senden. Um zu verhindern, dass der Master periodisch auf das Eintreffen einer Nachricht anfragen muss, kann der Interrupt

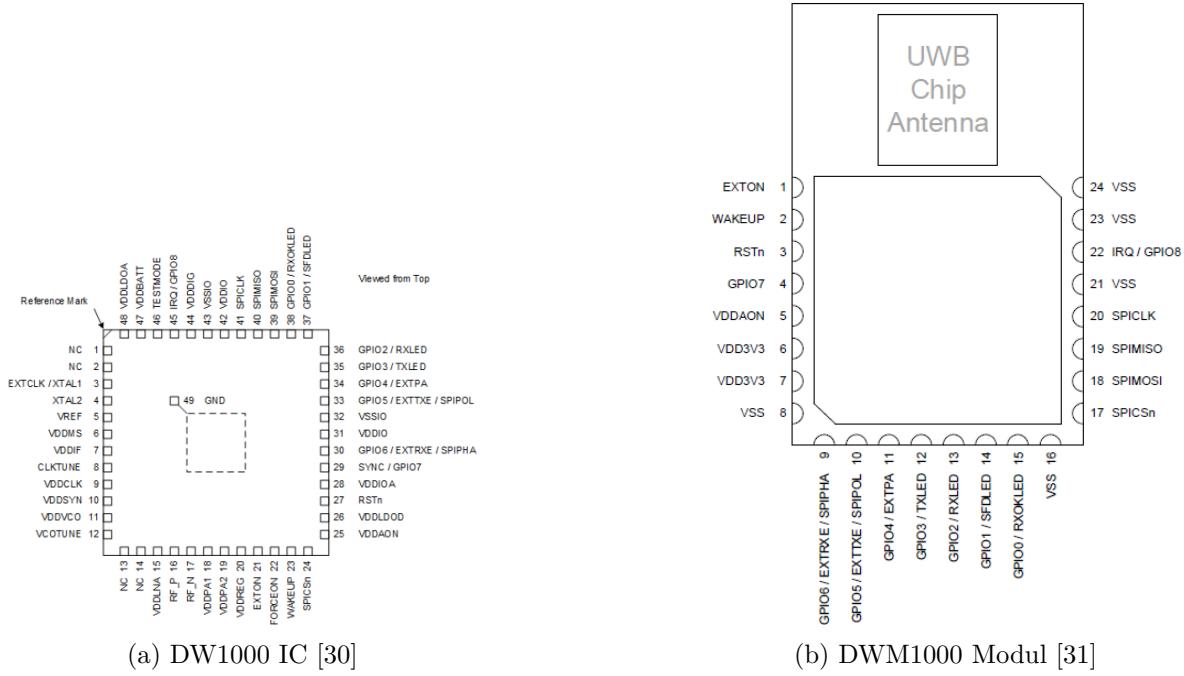


Abbildung 4.1.: DecaWave IC Pin Belegung

Request (IRQ)-Pin des Slaves verwendet werden. Um zu verhindern das kurzfristige Spannungsspitzen einen Interrupt auslösen, muss der IRQ-Pin über einen Pulldown-Widerstand auf Masse gezogen werden.

Um das DWM1000 erfolgreich zu initialisieren muss der RSTn-Pin durch den Mikrocontroller angesteuert werden. Zusätzlich ergibt sich, über die Beschaltung dieses Pins, die Möglichkeit den DWM1000 per Hardware im laufenden Betrieb neu zu starten.

Zusätzliche Informationen, wie der Versand und Empfang von Nachrichten, könnten über Statusleuchtdioden ausgegeben werden. Hierfür wird jeder der Pins GPIO1 bis GPIO3 jeweils mit einem Vorwiderstand und einer Leuchtdiode verbunden.

### Mikrocontroller

Wie bereits im Abschnitt 4.2.2 erwähnt beträgt die Logikpegelspannung 3,3 V. Durch diesen Umstand entfallen alle Mikrocontroller die mit einer 5 V Versorgungsspannung, wie z.B. der beliebte Arduino Uno, betrieben werden. Die Entscheidung fiel auf den Pro Trinket<sup>2</sup> der Firma Adafruit, der als Hauptprozessor den ATmega328/P verwendet, siehe Abbildung 4.2. Dieser hat den Vorteil, dass er jeweils in einer 5 V und 3,3 V Variante existiert. Zusätzlich ist die 3,3 V Variante mit einem Systemtakt von 12 MHz schneller als der vergleichbare Arduino Pro Mini 3,3 V der nur mit 8 MHz getaktet ist.

Um eine Kommunikationsverbindung zwischem dem DWM1000 und dem Mikrocontroller herzustellen, müssen die Pins anhand der Tabelle B.1 verbunden werden.

<sup>2</sup>Der Adafruit Pro Trinket 3,3 V ist zum Großteil pinkompatibel zu der 5 V Variante. Nur der Pin BAT+ benötigte eine Batteriespannung von 3,5 V–16 V und der drei Reihen weiter unten liegende 5 V Pin liefert nur 3,3 V.

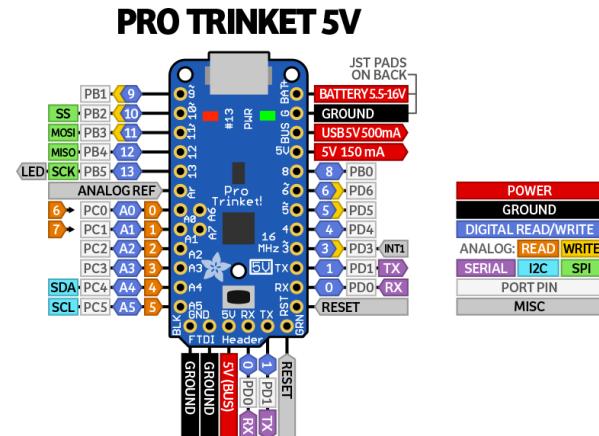
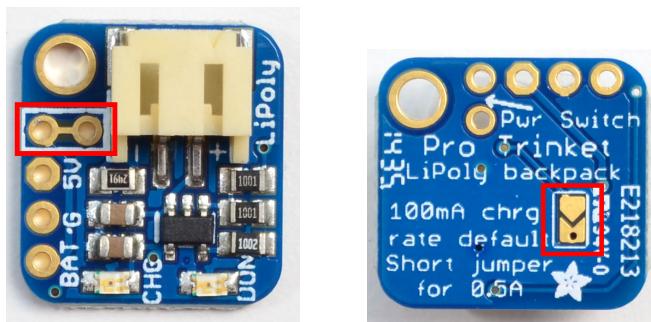


Abbildung 4.2.: Adafruit Pro Trinket [32]



(a) Modifikation für den Schalter.

(b) Modifikation für einen höheren Ladestrom.

Abbildung 4.3.: Adafruit Pro Trinket LiPoly/LiIon Backpack [33]

## Energieversorgung

Um den UWB Transceiver und den Pro Trinket mit Energie zu versorgen wird ein Lithium-Ionen Akku mit einer Spannung von 3,7 V und einer Kapazität von 2200 mAh verwendet. Die Verbindung zwischen den beiden wird über einen Lithium-Ionen Akku Lade-Chip hergestellt. Diesen gibt es als fertiges Modul von Adafruit mit der Bezeichnung Pro Trinket LiPoly/LiIon Backpack.

Bevor jedoch dieses Modul eingesetzt werden kann, müssen noch zwei Modifikationen durchgeführt werden. Zum einen kann die Energiequelle mittels eines Schalters vom Verbraucher getrennt werden. Per Standard sind jedoch diese zwei Pins mit einander verbunden und müssen mit einem scharfen Messer unterbrochen werden, siehe Abbildung 4.3a. Zum anderen wird der Lithium-Ionen Akku nur mit einem Strom von 100 mA geladen. Bei einer Kapazität von 2200 mAh würde ein vollständiger Ladezyklus ca. 22 h dauern. Um diese Zeit zu verkürzen, müssen die zwei Lötpads, siehe Abbildung 4.3b, miteinander verbunden werden. Danach wird der Lithium-Ionen Akku mit einem Strom von 500 mA geladen und dementsprechend verkürzt sich die Ladedauer auch auf ca. 2,5 h.

Um eine Verbindung zwischen dem Pro Trinket LiPoly/LiIon Backpack und dem Pro Trinket herzustellen, müssen die Pins anhand der Tabelle B.2 verbunden werden.

Um den Lithium-Ionen Akku zu laden, bedarf es nicht mehr als den Universal Serial Bus (USB) Stecker des Pro Trinket mit dem Computer oder einem USB Ladegerät zu verbinden.

### Datenaustausch

Der ATmega328/P verfügt nicht über einen eingebauten USB Controller, daher ist ein direkter Datenaustausch zwischen dem Mikrocontroller und der Verarbeitungseinheit nicht möglich. Jedoch verfügt der ATmega328/P über eine Universal Asynchronous Receiver Transmitter (UART) Schnittstelle, mit der Daten seriell über die Pins RX und TX übertragen und empfangen werden können. Mittels eines zusätzlichen Moduls kann diesen Datenstrom aufgefangen und über die USB Schnittstelle übertragen werden. Der CP2104 Friend von Adafruit erledigt genau diese Aufgabe. Angeschlossen wird er über den Future Technology Devices International (FTDI) Header, siehe Abbildung 4.2. Dadurch ist es möglich die UWB Module, die einen Datenaustausch benötigen, mit einem entsprechenden Modul auszurüsten.

Um eine Verbindung zwischen dem CP2104 Friend und dem Pro Trinket herzustellen, müssen die Pins anhand der Tabelle B.3 verbunden werden.

### Materialkosten pro UWB Modul

Die Materialkosten für einen Tag betragen jeweils 45,15 € und für einen Anker 56,24 €. Der Unterschied ergibt sich dadurch, dass z.B. für einen Tag kein Lithium-Ionen Akku benötigt wird, da dieser von der Energiequelle des Roboters gespeist wird. Die detaillierte Auflistung der Materialkosten können in der Tabelle B.4 nachgeschlagen werden. Die Bauelemente die jeweils für einen Tag bzw. Anker benötigt werden, sind in der dazugehörigen Spalte angekreuzt.

### Schaltplan

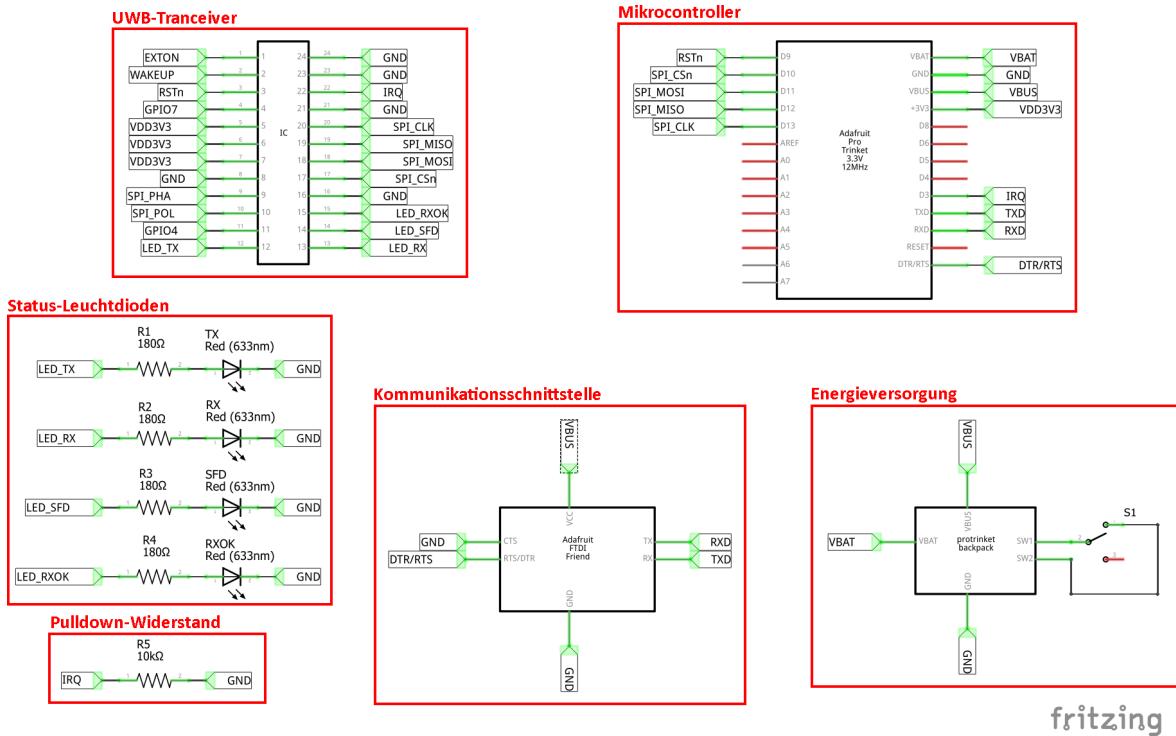
Eine Übersicht der besprochenen Bauelemente, gruppiert nach Funktionsgruppen, können dem Schaltplan in Abbildung 4.4 entnommen werden.

#### 4.2.3. Prototypen

Anhand des vorläufigen Schaltplans wurden zwei Prototypen erstellt, siehe Abbildung 4.5.

Der erste Prototyp wurde auf einer Steckplatine realisiert, siehe Abbildung 4.5a. Die elektrische Verbindung zu dem DWM1000 wurde dabei über eine Freiluftverdrahtung mit einem steckplatinenfähigen Adapter umgesetzt. Der erste Verbindungsauflaufbau wurde dabei mit dem Beispielprogramm BasicConnectivityTest aus dem GitHub Projekt [34] durchgeführt. Hierbei ist zu beachten das der PIN\_IRQ bei dem Pro Trinket nicht auf Pin 2 sondern auf Pin 3 liegt.

Während dem Aufbau des ersten Prototypen wurde ein Printed Circuit Board (PCB) Adapterboard, siehe Abbildung A.4, für den DWM1000 bestellt. Dieses PCB Adapterboard wurde ebenfalls dem GitHub Projekt [34] entnommen. Mit diesem wurde dann ein zweiter Prototyp aufgebaut, siehe Abbildung 4.5b, der zusätzlich ohne eine externe Energieversorgung betrieben werden konnte. Mittels der beiden Prototypen war ein erster Nachrichtenaus-



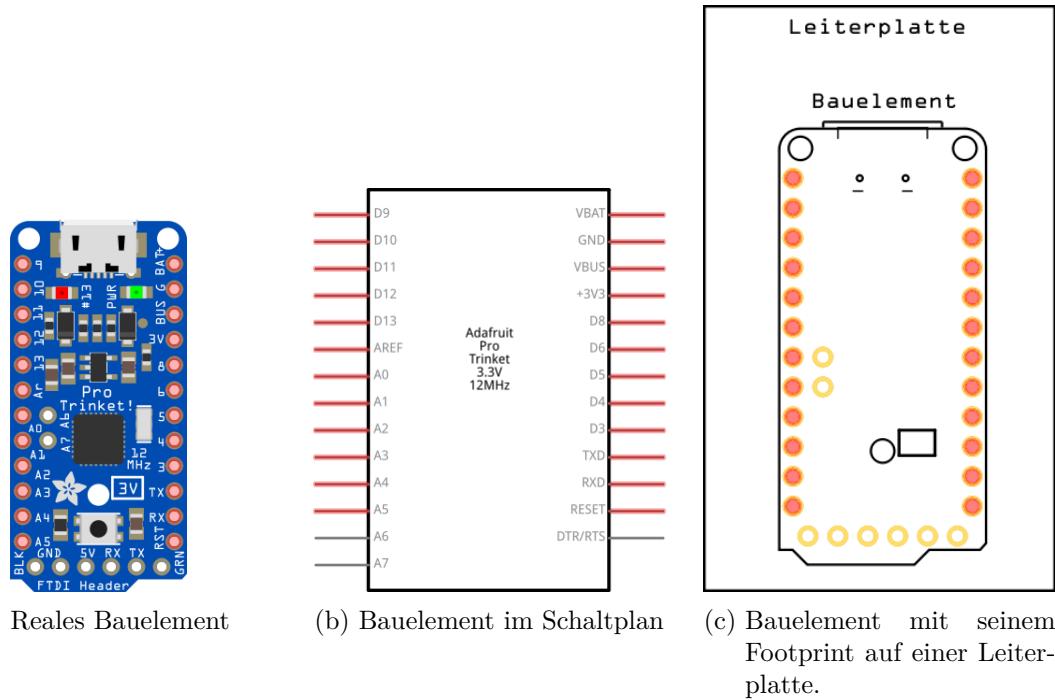


Abbildung 4.6.: Designstufen eines elektrischen Bauelementes.

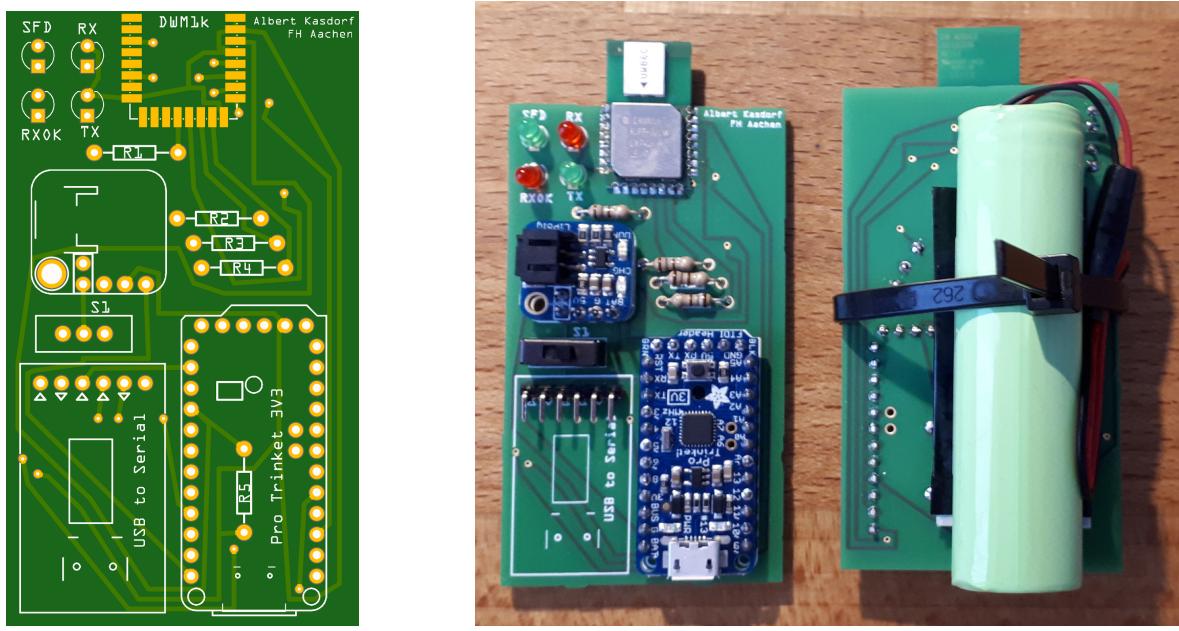
des Footprints muss nun jedes Bauteil so platziert werden, das es idealerweise möglichst wenig Platz auf der Leiterplatte verschwendet wird. Den je kleiner die Leiterplatte ist, desto günstiger in der Herstellung.

Die Leiterplatte besitzt eine rechteckige Form, mit den Maßen  $43 \times 75$  mm. Am höchsten Punkt der Leiterplatte befindet sich die Antenne des UWB Transceiver, dadurch wird gewährleistet das die Antenne in horizontaler Richtung ein gleichmäßiges Abstrahlmuster aufweist, siehe Abbildung A.5. Des Weiteren empfehlen die „Application Board Layout Guidelines“ das unterhalb der Antenne sich keine Metall oder andere nicht Radio Frequency (RF) transparente Materialien befinden und ein freier horizontaler Abstand von mindestens 10 mm eingehalten wird, siehe Abbildung A.6. [31]

Links neben dem UWB Transceiver befinden sich, gut sichtbar, die Statusleuchtdioden. Unterhalb der Statusleuchtdioden befindet sich der Ladeschaltkreis für den Lithium-Ion Akku sowie der Ein-Aus-Schalter. Auf der Rückseite der Leiterplatte wird der Lithium-Ion Akku befestigt. Dadurch ist ein aufrechter Stand des UWB Modul möglich. Die Leiterplatte ist mit 75 mm ca. 10 mm größer als der Lithium-Ion Akku, sodass es zu keiner Beeinträchtigung der Antenne kommt.

Im unteren rechten Bereich der Leiterplatte befindet sich der Pro Trinket, also die Verarbeitungseinheit. Direkt daneben auf der linken Seite ist das optionale Datenübertragungsmodul vorgesehen. Da der Tag auf dem Roboter in einer erhöhten Position befestigt wird, zeigt der USB Stecker für das Datenübertragungsmodul in eine ideale Richtung.

Nachdem jedes elektrische Bauelement auf der Leiterplatte positioniert wurde, muss noch die Verdrahtung der einzelnen Bauelemente erfolgen. Sehr bequem erfolgt die Verdrahtung über die Autoroute-Funktion. Hierbei errechnet ein Algorithmus, welchen Weg die Leiterbahnen nehmen sollen um alle notwendigen Pins miteinander zu Verbindungen ohne das sich ihre



(a) Das fertige Leiterplatten Layout.

(b) Vorder- und Rückansicht eines UWB Modul.

Abbildung 4.7.: Fertige UWB Module.

Wege überschneiden. Das fertige Layout der Leiterplatte ist in Abbildung 4.7a abgebildet. In der Abbildung 4.7b ist ein fertig aufgebautes UWB Modul abgebildet.

#### 4.2.5. Steuerosoftware

Als Basis für die Steuerosoftware wurde das GitHub Projekt [34] verwendet. Neben einer stabilen Basis für die Kommunikation mit dem DWM1000, existiert eine Implementierung des Two Way Ranging (TWR) Verfahrens von DecaWave. Die aktive Entwicklung ist mittlerweile eingestellt, es werden jedoch weiterhin Pull-Requests<sup>3</sup> für Fehlerkorrekturen und neue Funktionalitäten angenommen.

#### Klassenübersicht

Die Klasse DW1000 enthält alle Basisfunktionalität um mit dem DWM1000 arbeiten zu können. Dazu gehören die Kommunikation über den SPI Bus, das Lesen und Schreiben der Konfiguration und das Erstellen und Empfangen von Nachrichten. Über einen Low-Level Zugriff ist es zusätzlich möglich auf Funktionalitäten des DW1000 zuzugreifen die in dem ursprünglichen Klassendesign nicht vorgesehen waren.

Zeitstempel spielen beim Nachrichtenaustausch und bei der Entfernungsmessung eine zentrale Rolle. Jedoch werden die Zeitstempel als 40 bit Zahl gespeichert. Um die Verwendung zu erleichtern und Fehlerquellen auszuschließen wurde die Hilfsklasse DW1000Time bereitgestellt.

<sup>3</sup>Ein Pull-Request dient dazu, eine Weiterentwicklung oder Fehlerkorrektur mit anderen Entwicklern zu diskutieren und dann auf einfache Weise in den Hauptentwicklungszweig zu integrieren.

```
#include <SPI.h>
#include <DW1000Ranging.h>

void setup() {
    Serial.begin(115200);

    DW1000Ranging.initCommunication(9, SS, 3);

    DW1000Ranging.startAsAnchor("B1:00:00:00:B1:6B:00:B5",
        → DW1000.MODE_LONGDATA_RANGE_ACCURACY, false);
}

void loop() { DW1000Ranging.loop(); }
```

Auflistung 4.1: Quellcode für ein UWB Modul das als Anker konfiguriert ist.

Über die Klasse DW1000Ranging ist eine Implementierung des TWR Verfahrens von Deca-Wave verfügbar, siehe Abschnitt 2.1.2. Mittels drei Rückruffunktionen (engl. Callback function) teilt die Klasse dem Anwender mit, welche Entfernung gemessen wurde und ob neue Quellen für Entfernungsmessungen hinzugekommen oder verschwunden sind. Jede Quelle für eine Entfernungsmessung wird mit der Klasse DW1000Device modelliert. Neben der gemessenen Entfernung beinhaltet die Klasse auch eine eindeutige Identifikationsnummer.

## Datenaustausch

Der erste Bereich des Datenaustausches findet zwischen den UWB Modulen statt. Hierfür wird mindestens ein Anker und Tag benötigt. In der Auflistung 4.1 ist der Quellcode eines Anker abgebildet. Über die ersten zwei Zeile wird die SPI- und DW1000Ranging-Bibliothek in das Programm eingebunden. Um eine Kommunikation mit dem DWM1000 aufzubauen muss der Methode initCommunication die Belegung des Reset-, SS- und IRQ-Pin übergeben werden. Nach dem Aufruf der Methode startAsAnchor fungiert das UWB Modul als Anker. Die notwendigen Parameter sind dabei die eindeutige Adresse, der Messmodus und ob eine zufällige Geräteadresse generiert werden soll. Wenn der letzte Parameter false ist, werden aus den ersten zwei Bytes der eindeutigen Adresse, die Geräteadresse gebildet. Der zyklische Aufruf der Methode loop sorgt für die Nachrichtenverarbeitung des UWB Moduls.

Das Gegenstück zum Anker ist in der Auflistung 4.2 abgebildet. Jede gemessene Entfernung zwischen dem Tag und einem Anker soll per USB an den Computer übertragen werden. Der Datenaustausch erfolgt dabei im JavaScript Object Notation (JSON) Format, daher wird die Bibliothek ArduinoJson benötigt.

Da dieses UWB Modul als Tag fungieren soll, muss anstatt startAsAnchor die Methode startAsTag aufgerufen werden. Die Bedeutung der Parameter zwischen den beiden Methoden bleibt identisch.

Neu ist der Aufruf der Methode attachNewRange, attachNewDevice und attachInactiveDevice. Jeder der Methoden wird die Adresse einer Rückruffunktion übergeben. Diese Funk-

tionen werden immer dann aufgerufen, wenn eine neue Entfernung gemessen wurde und wenn eine neue Quelle für die Entfernungsmessungen hinzugekommen oder verschwunden ist.

Alle Rückruffunktionen rufen ihrerseits die Funktion `transmitDATA` auf. In dieser wird mittels der Klasse `StaticJsonBuffer` ein `JsonObject` erstellt, mit Daten im Schlüssel-Wert-Format (engl. Key-Value) belegt und dann mittels der Methode `printTo(Serial)` über die USB Schnittstelle an den Computer versendet. Das Format der Daten ist in der Auflistung 4.3 abgebildet.

Nachdem die Daten an den Computer gesendet wurden, müssen diese entgegengenommen und in das ROS System eingespeist werden. Dies geschieht über die Auflistung 4.4.

Zuerst wird das ROS System über `rospy.init_node` initialisiert. Um Nachrichten versenden zu können, wird zusätzlich ein Publisher benötigt. Dieser wird über den Aufruf von `rospy.Publisher` erstellt. Die Verbindung zu dem UWB Modul wird über den Aufruf `serial.Serial` hergestellt.

Mittels der ROS Nachricht `ObservationRangeBeacon` wird die Sensor-Charakteristik des UWB Moduls beschrieben. Hierzu gehören die minimal und maximal messbare Entfernung, das Sensorrauschen, eine oder mehrere Entfernungsmessungen und einige ROS spezifische Angaben wie z.B. das Koordinatensystem in dem sich das UWB Modul befindet.

Danach werden solange Daten von dem UWB Modul gelesen bis der ROS Knoten beendet wird. Jeder gelesene Datensatz wird mittels der Methode `json.loads(line)` untersucht, in ein JSON Objekt konvertiert, und im Fehlerfall übersprungen. Gültige Entfernungsdaten werden in den Attributen `range` und `id` des Feldes `orb.sensed_data` gespeichert.

Der Aufruf `pub.publish(orb)` übergibt die zuvor erstellte Nachricht an das ROS System.

Der vollständige Quellcode mit Kommentaren und Fehlerbehandlung kann in dem GitHub Projekt [36] eingesehen werden.

#### 4.2.6. Kalibrierung

Je genauer sich die Zeitstempel für den Empfang bzw. für das Versenden einer Nachricht bestimmen lassen, desto genauer ist auch die gemessene Entfernung. Idealerweise entspricht die Differenz zwischen zwei Zeitstempeln der ToF. In der Realität ist die Differenz jedoch deutlich größer. Als eine der Ursachen kann die herstellungsbedingte Ungenauigkeiten in den verwendeten Komponenten in Betracht gezogen werden, genauer gesagt geht es um die Antennenverzögerung. Unter der Antennenverzögerung versteht man die Zeitspanne die die Antenne benötigt um mit der Übertragung der Nachricht zu starten bzw. um eine Nachricht zu empfangen, siehe Abbildung 4.8.

Die gemessene Zeitspanne ergibt sich dabei aus der Gleichung:

$$tof_{measured} = t_{ADTX} + tof_{actual} + t_{ADRX} \quad (4.1)$$

wobei der Term  $t_{ADTX}$  die Antennenverzögerung für das Versenden und  $t_{ADRX}$  für das Empfangen auf den gegenüberliegenden UWB Modulen beschreibt. Es wird davon ausgegangen, dass die Antennenverzögerung für das Senden und Empfangen auf einem UWB Modul gleich lang ist. [8]

```

#include <SPI.h>
#include <DW1000Ranging.h>
#include <ArduinoJson.h>

enum class Events : byte {
    NEW_RANGE=0, NEW_DEVICE=1, INACTIVE_DEVICE=2
};

void setup() {
    Serial.begin(115200);

    DW1000Ranging.initCommunication(9, SS, 3);
    DW1000Ranging.attachNewRange(newRange);
    DW1000Ranging.attachNewDevice(newDevice);
    DW1000Ranging.attachInactiveDevice(inactiveDevice);

    DW1000Ranging.startAsTag("A0:00:00:00:B1:6B:00:B5",
        → DW1000.MODE_LONGDATA_RANGE_ACCURACY, false);
}

void loop() {
    DW1000Ranging.loop();
}

void newRange() {
    transmitDATA(Events::NEW_RANGE, DW1000Ranging.getDistantDevice());
}

void newDevice(DW1000Device* device) {
    transmitDATA(Events::NEW_DEVICE, device);
}

void inactiveDevice(DW1000Device* device) {
    transmitDATA(Events::INACTIVE_DEVICE, device);
}

void transmitDATA(const Events event, const DW1000Device* device) {
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject& root = jsonBuffer.createObject();

    root["type"] = static_cast<byte>(event);
    root["aa"] = device->getShortAddress();
    root["r"] = (event==Events::NEW_RANGE) ? device->getRange() : 0.0f;

    root.printTo(Serial);
    Serial.println();
}

```

Auflistung 4.2: Quellcode für ein UWB Modul das als Tag konfiguriert ist.

```
{"type": 0, "aa": 160, "r": 1.01}
```

Auflistung 4.3: Datenzeile im JSON Format beim Austausch zwischen UWB Modul und Computer.

```
import json
import rospy
import serial

from mrpt_msgs.msg import ObservationRangeBeacon
from mrpt_msgs.msg import SingleRangeBeaconObservation

def main():
    rospy.init_node('beacon_publisher')
    pub = rospy.Publisher('/beacon', ObservationRangeBeacon, queue_size=10)
    ser = serial.Serial('/dev/CP2104_Friend', 115200, timeout=2)

    orb = ObservationRangeBeacon()
    orb.header.frame_id = 'uwb_reciever_link'
    orb.min_sensor_distance = 0.0
    orb.max_sensor_distance = 120.0
    orb.sensor_std_range = 0.1
    orb.sensed_data.append(SingleRangeBeaconObservation())

    while not rospy.is_shutdown():
        line = ser.readline()
        try:
            obj = json.loads(line)
        except ValueError:
            continue

        orb.header.stamp = rospy.Time.now()
        orb.header.seq = orb.header.seq + 1
        orb.sensed_data[0].range = obj["r"]
        orb.sensed_data[0].id = obj["aa"]

    pub.publish(orb)
```

Auflistung 4.4: Quellcode um eine Entfernungsmessung an das ROS System zu übergeben.

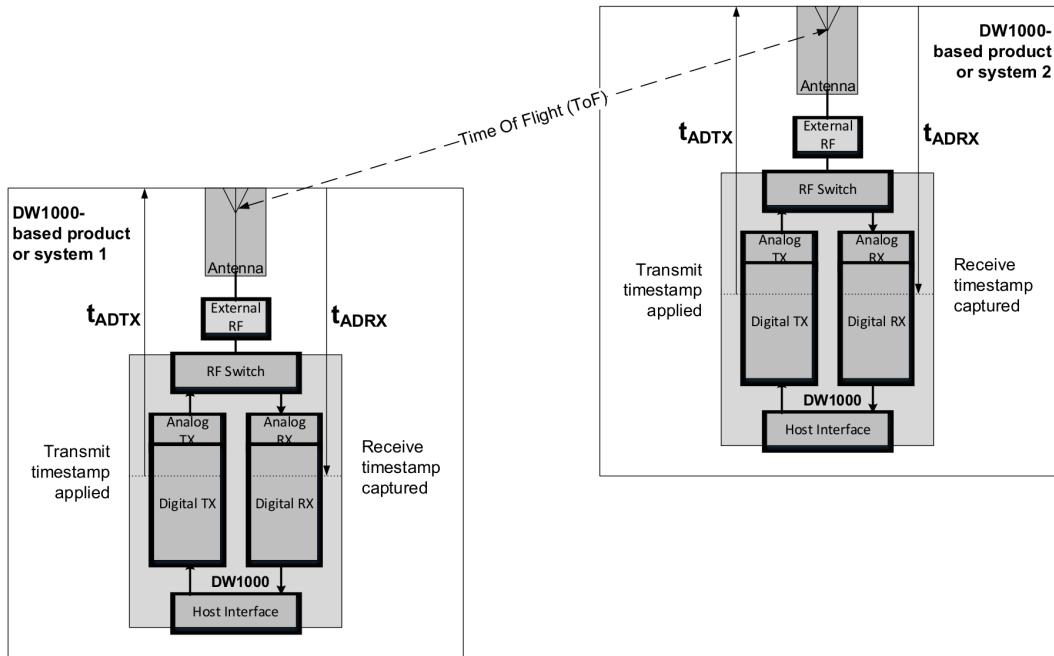


Abbildung 4.8.: Antennenverzögerung zwischen zwei UWB Modulen. [37]

Um die Antennenverzögerung zu bestimmen, wird diese initial auf null eingestellt, die UWB Modul in einem bekannten Abstand positioniert und dann von jedem UWB Modul eine Reihe von Messungen aufgezeichnet. In den nachfolgenden Abschnitten werden zwei Verfahren vorgestellt, mit der sich die Antennenverzögerungen aus den aufgezeichneten Messungen bestimmen lässt.

### DecaWave

Um die Antennenverzögerung pro UWB Modul zu bestimmen, wird von DecaWave ein generischer Algorithmus vorgeschlagen. [37]

Im ersten Schritt wird eine initiale Kandidatenliste mit eintausend Kandidaten erstellt. Jeder Kandidat beinhaltet für jedes UWB Modul eine Antennenverzögerungen. Die Werte sind dabei gleichmäßig um den Wert 513 ns mit einer Streuung von 6 ns verteilt.

Im nächsten Schritt wird für jeden Kandidaten seine spezifische ToF Zeit berechnet, siehe Gleichung 4.2.

$$tof_{candidate} = \frac{1}{2}del_{chip1} + \frac{1}{2}del_{chip2} + tof_{measured} \quad (4.2)$$

Bei  $tof_{measured}$  handelt es sich um eine  $n \times n$  Matrix mit den gemessenen ToF Zeiten zwischen den UWB Modulen, siehe Gleichung 4.3.

$$tof_{measured} = \begin{pmatrix} 0 & tof_{m12} & tof_{m13} \\ tof_{m21} & 0 & tof_{m23} \\ tof_{m31} & tof_{m32} & 0 \end{pmatrix} \quad (4.3)$$

Die Matrix  $del_{chip1}$  enthält pro Spalte die Antennenverzögerung eines UWB Moduls und die diagonalen Element sind alle null. Bei der Matrix  $del_{chip2}$  handelt es sich um die transponierte Matrix von  $del_{chip1}$ , siehe Gleichung 4.4.

$$Del_{chip1} = \begin{pmatrix} 0 & t_{ad2} & t_{ad3} \\ t_{ad1} & 0 & t_{ad3} \\ t_{ad1} & t_{ad2} & 0 \end{pmatrix}; \quad Del_{chip2} = \begin{pmatrix} 0 & t_{ad1} & t_{ad1} \\ t_{ad2} & 0 & t_{ad2} \\ t_{ad3} & t_{ad3} & 0 \end{pmatrix} \quad (4.4)$$

Die Bewertung eines Kandidaten findet dabei über die Berechnung der Norm aus der Differenz zwischen der tatsächlichen ToF Zeit und ToF Zeit des Kandidaten statt, siehe Gleichung 4.5. Je kleiner die Norm desto besser die Bewertung des Kandidaten.

$$rank_{candidate} = \|(tof_{actual} - tof_{candidate})\| \quad (4.5)$$

Nach dem alle Kandidaten bewertet worden sind, werden die besten 25 % weiterverwendet und die Restlichen verworfen. Um die Kandidatenliste nicht mit jeder Iteration kleiner werden zu lassen, werden aus den verbliebenen Kandidaten drei neue Gruppen mit einer alle zwanzig Iterationen abnehmenden Streuung erstellt.

Der Vorgang der Erstellung und Bewertung einer Kandidatenliste wird hundert Mal durchgeführt und sollte sich zum Ende hin der tatsächlichen Antennenverzögerung annähern.

### Lineares Gleichungssystem

Bei zwei UWB Modulen kann die Gleichung 4.1 zwei Mal aufgestellt werden, für jede Richtung einmal. Bei drei UWB Modulen bereits sechs Mal. Allgemein gilt für die Anzahl der Gleichung die Formel  $n(n - 1)$ . Damit lässt sich ein lineares Gleichungssystem aufstellen und somit die Antennenverzögerung pro UWB Modul berechnen.

Hierfür werden alle unbekannten Terme auf die linke Seite gebracht und alle bekannten auf die rechte Seite, siehe Gleichung 4.6.

$$t_{ad1} + t_{ad2} = tof_{measured} - tof_{actual} \quad (4.6)$$

Danach werden alle Gleichung in die Matrixform  $A \cdot x = b$  gebracht und mit einem beliebigen Lösungsverfahren gelöst, siehe Gleichung 4.7 für drei UWB Module. Die Koeffizientenmatrix enthält hierbei pro Zeile immer nur zwei Einsen, da für jede Entfernungsmessung nur zwei Antennenverzögerungen berücksichtigt werden.

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} t_{ad1} \\ t_{ad2} \\ t_{ad3} \end{pmatrix} = \begin{pmatrix} tof_{measured1} - tof_{actual1} \\ tof_{measured2} - tof_{actual2} \\ tof_{measured3} - tof_{actual3} \end{pmatrix} \quad (4.7)$$



# 5.

---

## Umsetzung des RO-SLAM in ROS

---

Im vorherigen Kapitel wurden mehrere UWB Modul erstellt, darunter befand sich ein Tag und vier Anker. Um die Entferungen messen zu können, muss der Tag auf einer Roboterplattform montiert und die Anker im Messbereich verteilt werden. Damit ist aber nur ein Teil der Hardwareanforderungen erfüllt. Es wird noch eine Roboterplattform benötigt, die zum einen um den Tag erweitert werden kann und zum anderen über ausreichend Leistungsreserven verfügt um alle ROS Module ausführen zu können, die für einen RO-SLAM benötigt werden. Die Zusammenstellung dieser Roboterplattform ist bestandteil dieses Kapitel.

Die eingesetzte Hardware bildet einen soliden Sockel für die Ausführung der ROS Module. Erweitert werden muss dieser noch um die Softwareseite. Mit der Übersicht über die Softwarearchitektur lässt sich eine grobe Vorstellung der eingesetzten ROS Module vermitteln. Einen tieferen Einblick in die ROS Module liefern die Abschnitte mit den ROS Hauptmodulen und -Nebenmodulen. Bei dem ersten geht es um die Kernmodule die eine direkte bzw. indirekte Voraussetzung für den RO-SLAM sind. Im letzteren werden die Module behandeln die notwendigen sind um zusätzliche Daten, Kennzahlen und Grafiken für die Auswertung zu extrahieren.

Nach dem die Hardware- und Softwarevoraussetzungen für den RO-SLAM Algorithmus erfüllt wurden, wird ein Blick auf die Umsetzung des RO-SLAM in dem MRPT Framework geworfen.

### 5.1. Roboterplattform

Um die Messungen für den RO-SLAM aufzuzeichnen wird eine Roboterplattform benötigt, auf der das UWB Modul montiert werden kann. Zusätzlich wird noch eine Reihe weiterer Sensoren benötigt, die im Folgenden vorgestellt werden.

Also Roboterplattform dient dabei der Robotino 2 von Festo Didactic. Er gehört zur Klasse der holonomen Roboter, d.h. seinen Bewegungsmöglichkeiten unterliegen keinerlei Einschränkungen im zweidimensionalen Raum. Ermöglicht wird das, durch drei voneinander unabhängig arbeitenden omnididirektionalen Antriebseinheiten. Jede der drei Antriebseinheiten verfügt über einen Inkrementalgeber, mit dessen Hilfe abgeschätzt wird, wie weit der Robotino gefahren ist. Um den Fehler der Inkrementalgeber in Kurvenfahrten zu reduzieren wird das digitale Gyroskop<sup>1</sup> von Microinfinity eingesetzt. Gesteuert wird der Robotino über eine 300 MHz star-

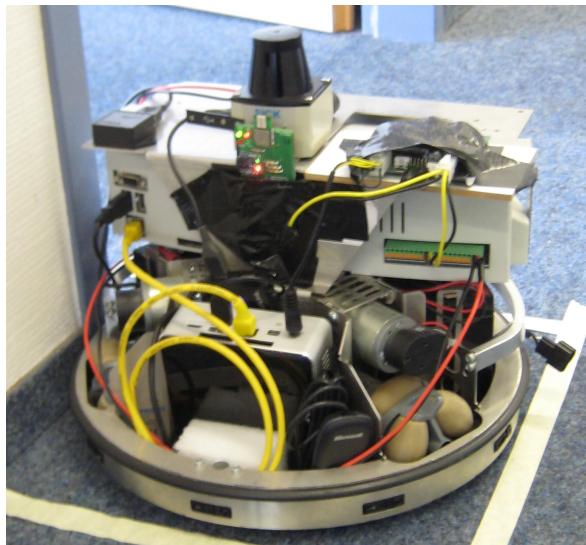


Abbildung 5.1.: Die Frontansicht der fertig aufgebauten Roboterplattform.

ke Verarbeitungseinheit auf Basis eines Linux Betriebssystems mit einem Echtzeitkernel. [38]

An der höchsten Position des Robotinos befindet sich ein 2D-Laser-Entfernungsmeßgeräte<sup>2</sup> der Firma Sick. Mit einem Öffnungswinkel von 270° und einem Arbeitsbereich von 0,05 m–25 m ist es möglich, genaue Belegtheitskarten der Umwelt zu erstellen. [39]

Die Verarbeitungseinheit des Robotinos ist leistungsfähig genug um die anstehenden Steuerungsaufgaben zu erfüllen. Jedoch ist diese von leistungshungrigen Anwendungen wie dem RO-SLAM überfordert. Aus diesem Grund wird eine zusätzliche Verarbeitungseinheit<sup>3</sup> mit einem Intel Core i7<sup>4</sup> verwendet.

Die Fahrbefehle werden dem Robotino über einen Xbox Wireless Controller übermittelt.

## 5.2. Softwarearchitektur

Der Softwarearchitektur, siehe Abbildung 5.2, lassen sich zwei Nachrichtentypen entnehmen die für den RO-SLAM essentiell sind. Zum einen sind das die Daten der Entfernungsmeßungen (mrpt\_msgs/ObservationRangeBeacon) zwischen den UWB Modul und dem Transformationsbaum (tf2\_msgs/TFMessage) auf der anderen. Die Nachrichten des Transformationsbaums werden dabei nur indirekt benötigt, um zu bestimmen wie weit sich die Roboterplattform bewegt hat, sprich die Odometrieinformationen.

Als Datenquelle für den Transformationsbaum sind zuerst die statischen Transformationen zu nennen, dann die Belegtheitskarten für die Transformation von dem Odometrie- in das Kartenkoordinatensystem und zuletzt die zwei Arten der Odometrie. Die erste liefert die Daten anhand der Inkrementalgeber in der Antriebseinheit und die zweite eine Odometrie über das auswerten der 2D-Laser-Entfernungsmeßgeräts. Bei der Laser-Odometrie haben sich

<sup>1</sup>Model: CruzCore XG1000 / XG1010

<sup>2</sup>Model: TiM571-2050101

<sup>3</sup>Model: NUC5i7RYH

<sup>4</sup>5. Generation des Intel Core i7-5557U Prozessor mit bis zu 3,4 GHz. [40]

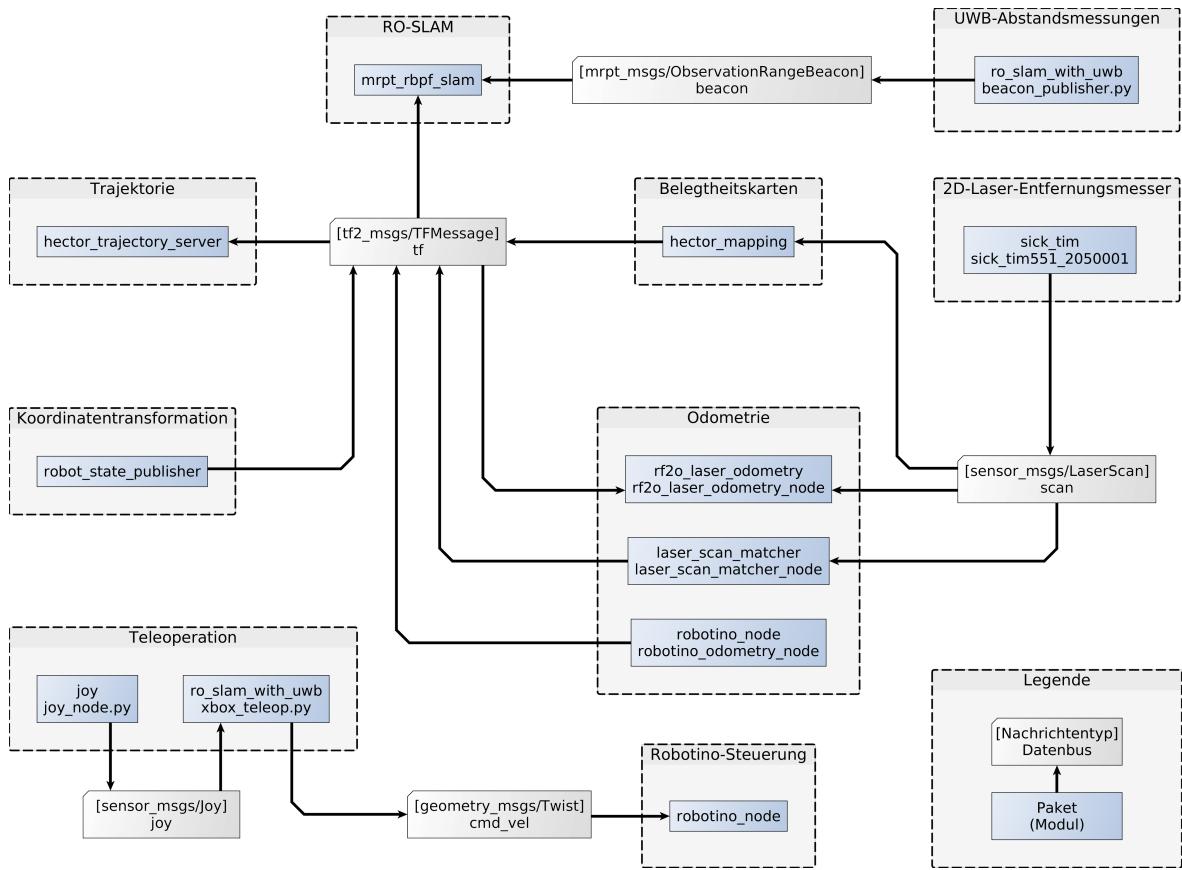


Abbildung 5.2.: Softwarearchitektur der ROS Module für den RO-SLAM.

zwei unterschiedliche Verfahren etabliert und alle drei Verfahren sollen auf ihre Genauigkeit untersucht werden.

Der untere Bereich, in der Abbildung 5.2, ist über keinen Datenbus mit den anderen ROS Modul verbunden. Jedoch wird dieser zum einen benötigt um die Kommunikation zwischen der Robotino Steuereinheit herzustellen und zum anderem um die Roboterplattform als Ganzes durch ein Eingabegerät zu steuern.

### 5.2.1. ROS Hauptmodule

#### Robotino-Steuerung

Zur Steuerung der Roboterplattform muss die Verarbeitungseinheit Befehle zur Steuereinheit des Robotinos schicken. Diese Aufgabe erfüllt das ROS Modul `robotino_node` aus dem Paket `robotino_node`. Über den Datenbus `cmd_vel` können die Geschwindigkeiten in X-/Y-Richtung und um die Z-Achse festgelegt werden. Um den Robotino zu stoppen, müssen alle drei Parameter auf den Wert null gesetzt werden.

Um an die Informationen der Inkrementalgeber zu kommen, muss das ROS Modul `robotino_node` aus dem Paket `robotino_node` gestartet werden. Nach dem Start steht der Datenbus `odom` mit den aktuellen Werten der Inkrementalgeber zur Verfügung. Zusätzlich sorgt dieses Modul auch für die dynamische Transformation zwischen dem `odom` und `base_link` Koordinatensystem.

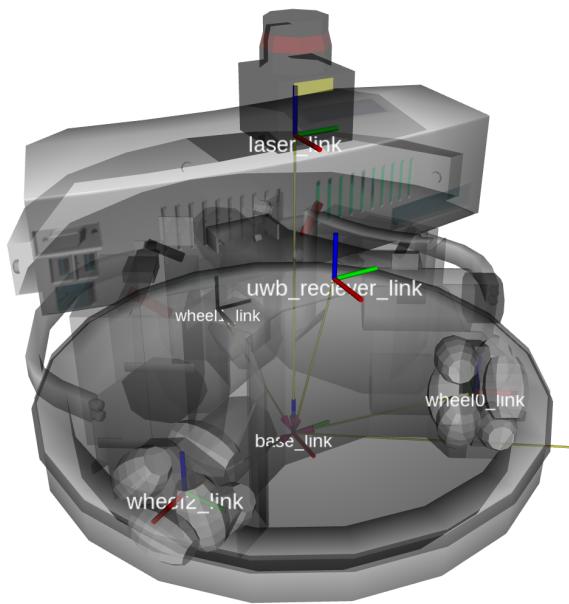


Abbildung 5.3.: Statische Transformationen der Roboterplattform.

### Koordinatentransformation

Neben den dynamischen Transformationen besteht die Roboterplattform auch aus vielen statischen Transformationen. Hierzu gehört z.B. die Transformation vom Mittelpunkt der Roboterplattform zum 2D-Laser-Entfernungsmesser oder UWB Modul. Diese statischen Transformationen werden in einer URDF Datei einmalig gespeichert und dann über das ROS Modul state\_publisher aus dem Paket robot\_state\_publisher zur Laufzeit bereitgestellt, siehe Abbildung 5.3.

### Teleoperation

Zur Steuerung der Roboterplattform wird ein Xbox Wireless Controller als Eingabegerät verwendet. Zustandsänderungen werden über das ROS Modul joy\_node aus dem Paket joy registriert und über den Datenbus joy bereitgestellt. Das überwachte Eingabegerät wird über den Parameter dev festgelegt.

Mit den generischen Daten des Eingabegerätes kann die Roboterplattform nichts anfangen, diese benötigt eine Nachricht vom Typ geometry\_msgs/Twist. Die Transformation zwischen der Nachricht sensor\_msgs/Joy und geometry\_msgs/Twist erfolgt durch das ROS Modul xbox\_teleop.py aus dem Paket ro\_slam\_with\_uwb.

### UWB Entfernungsmessung

Die Entfernungsmessung zwischen dem Tag und den vorhandenen Ankern werden durch das UWB Modul über die USB Schnittstelle bereitgestellt. Die Daten werden dann von dem ROS Modul beacon\_publisher.py aus dem Paket ro\_slam\_with\_uwb aufgefangen und auf den Datenbussen beacon und beacon\_raw veröffentlicht. Der erste Datenbus ist vom Typ mrpt\_msgs/ObservationRangeBeacon und wird von dem MRPT Modul für den RO-SLAM

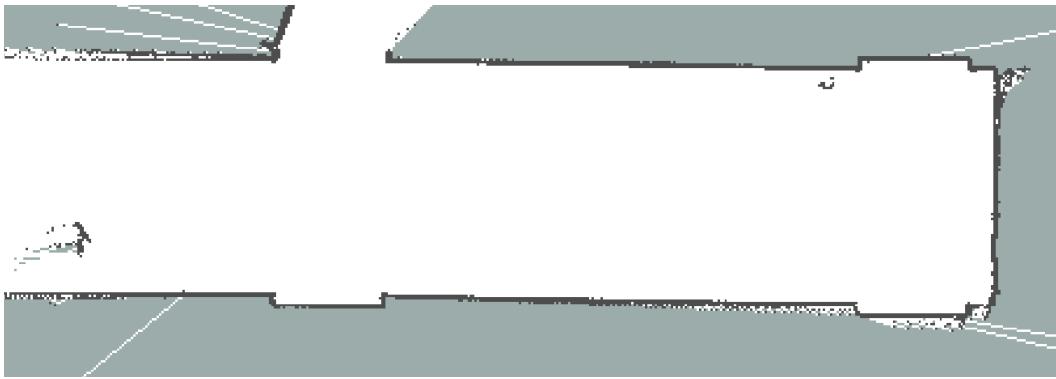


Abbildung 5.4.: Belegtheitskarte der Messstrecke.

benötigt. Der zweite Datenbus wird für die Auswertung der Entfernungsmessung im Kapitel 6 benötigt und ist vom Typ ro\_slam\_with\_uwb/Beacon.

### 5.2.2. ROS Hilfsmodule

#### 2D-Laser-Entfernungsmesser

Der 2D-Laser-Entfernungsmesser gehört zu einem der am häufigsten verwendeten Sensoren für die präzise visuelle Erfassen der Umwelt. Für mehrere der nachfolgenden ROS Modul ist eine 2D-Laser-Entfernungsmessung eine zwingende Voraussetzung. Um die Daten dem ROS System bereitzustellen wird das ROS Modul sick\_tim551\_2050001 aus dem Paket sick\_tim verwendet. Dieses Modul stellt über den Datenbus scan, mit einer Rate von 15 Hz, Nachrichten vom Typ sensor\_msgs/LaserScan bereit.

#### Belegtheitskarten

Das erste ROS Modul, das von der 2D-Laser-Entfernungsmessung Gebrauch macht, ist das hector\_mapping aus dem gleichnamigen Paket. Mit diesem Modul können präzise Belegtheitskarten (engl. Occupancy Grid Map) erstellt werden, die für jede Koordinate der Karte festlegen ob diese durch ein Hindernis belegt (schwarz) oder frei (weiß) ist. Bereiche die von Hindernissen verdeckt sind oder außerhalb der Sensorreichweite liegen, werden als unbestimmt (grau) markiert, siehe Abbildung 5.4. In ROS werden Belegtheitskarten mit dem Nachrichtentyp nav\_msgs/OccupancyGrid repräsentiert.

Über die minimale und maximale Sensorreichweite (laser\_min\_dist bzw. laser\_max\_dist) wird festgelegt, welche Messungen für die Kartenerstellung berücksichtigt werden. Es ist sinnvoll die Herstellerangaben von 0,05 m und 25,0 m zu verwenden.

Die Auflösung einer Zelle in der Karte wird über den Parameter (map\_resolution) festgelegt. Es eine Auflösung von 0,02 m pro Zelle verwendet.

#### Trajektorie

Neben den Entfernungsinformationen wertet der RO-SLAM auch die Daten der Odometrie aus. Als Quelle für die Odometrie dienen dabei die Inkrementalgeber der Roboterplattform. Im nächsten Abschnitt werden weitere Quellen für die Odometrie vorgestellt. Um die Güte

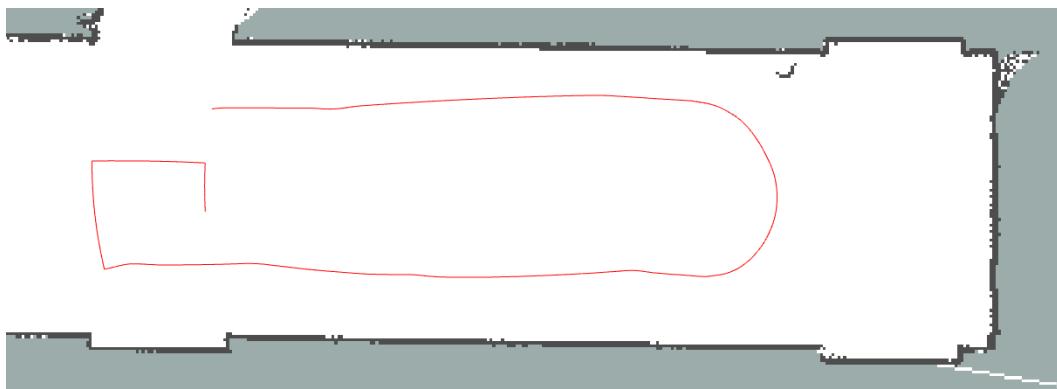


Abbildung 5.5.: Trajektorie einer Messfahrt der Roboterplattform.

der Odometrie-Quellen zu vergleichen, wird die verfahrene Trajektorie jeder Quelle benötigt. Diese Informationen werden von dem ROS Modul `hector_trajectory_server` aus dem gleichnamigen Paket bereitgestellt. Die Daten liegen dabei als Nachricht vom Typ `nav_msgs/Path` bereit und können in die Belegtheitskarte integriert werden, siehe Abbildung 5.5.

### Laser-Odometrie

Wie bereits im vorherigen Abschnitt erwähnt, werden hier zwei weitere Quellen für die Odometrie vorgestellt. Beide ROS Module nutzen nur die Daten der 2D-Laser-Entfernungsmessung, um eine Schätzung für die aktuelle Pose zu erstellen.

Bei dem ersten ROS Modul handelt es sich um den `laser_scan_matcher_node` aus dem Paket `laser_scan_matcher`. Hierbei handelt es sich um eine Variante des Iterative Closest Point (ICP) Verfahrens, welches versucht eine Transformation zwischen den Punkt einer Oberfläche aus den vorherigen und aktuellen Sensordaten zu finden. Jedoch wird anstatt einer Punkt-zu-Punkt- eine Punkt-zu-Linie-Metrik für die Minimierung verwendet. [41] Dieses Verfahren wird im Folgenden nur noch als LSM Verfahren bezeichnet.

Bei dem zweiten ROS Modul handelt es sich um den `rf2o_laser_odometry_node` aus dem Paket `rf2o_laser_odometry`. Hierbei handelt es sich um eine Implementierung des Range Flow-based 2D Odometry (RF2O) Verfahrens, bei dem jeder beobachtete Punkt des Sensors als eine Funktion der Geschwindigkeit des Sensors modelliert wird. Die Bewegung des Sensors wird dann aus der Minimierung der Dichteformulierung aller Punkt-Funktionen berechnet. [42]

### 5.2.3. MRPT

Bei dem MRPT Framework handelt es sich um eine Bibliothek die Datenstrukturen und Algorithmen aus dem aktiven Forschungsbereich der Robotik bereitstellt. Dadurch ist eine schnelle Umsetzung neuer Algorithmen, sowie der Vergleich mit bereits bestehenden möglich. Zu den verfügbaren Datenstrukturen zählen die Basisdatentypen der linearen Algebra bis hin zu den komplexeren Datentypen der Robotik wie z.B. die Belegtheitskarten. Zur direkten Verwendung existieren viele Algorithmen aus den Bereichen der SLAM-Algorithmen, des

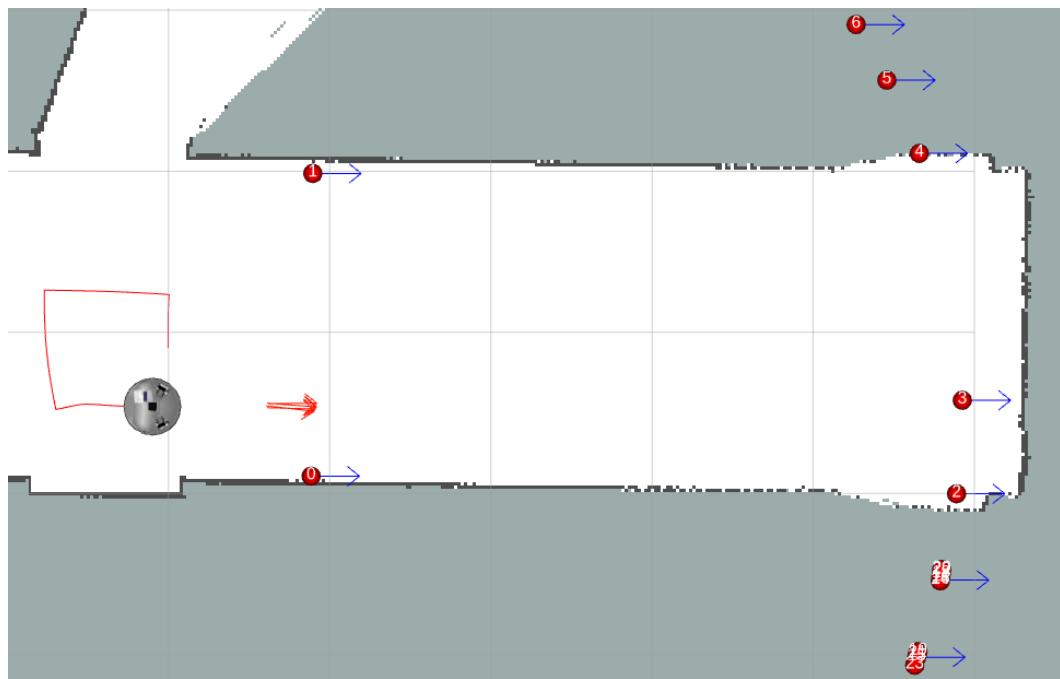


Abbildung 5.6.: Positionsschätzung für die Roboterplattform mit roten Pfeilen und für die UWB Module mit blauen Pfeilen.

Maschinellen Sehen (engl. Computer Vision) und der Pfad- und Bewegungsplanung (engl. Path/Motion Planning).

Aus dem Bereich der RO-SLAM Algorithmen sind die Verfahren aus den Veröffentlichungen „A pure probabilistic approach to range-only SLAM“ und „Efficient probabilistic range-only SLAM“ implementiert, siehe Abschnitt 3.3 und 3.4.

Um die RO-SLAM Algorithmen aus dem MRPT Framework zu nutzen, wird das ROS Modul mrpt\_rbpf\_slam aus dem Paket mrpt\_rbpf\_slam benötigt. Mit dem Parameter ini\_filename wird der Dateipfad zu der Konfigurationsdatei für die RO-SLAM Algorithmen angegeben und über den sensor\_source Parameter werden die Datenquellen für den RO-SLAM definiert. Hierbei ist es möglich sowohl die Entfernungsmessungen der UWB Modul als auch die der 2D-Laser-Entfernungsmesser anzugeben. Von der letzten Datenquelle wird kein Gebrauch gemacht, da diese Transformationen veröffentlicht, die mit den bereits bestehenden Koordinatensystemen kollidiert. Über die Datenbusse particlecloud und particlecloud\_beacons werden die Positionsschätzungen für die Roboterplattform und die UWB Modul veröffentlicht, siehe Abbildung 5.6.

Nach der Konfiguration der ROS Seite müssen noch Einstellungen für den RO-SLAM festgelegt werden. Hier muss die Konfigurationsdatei, die in dem ROS Parameter ini\_filename hinterlegt ist, bearbeitet werden. Das MRPT Framework verfügt über einen eigenen Abspielmechanismus für aufgezeichnete Messungen. Dieser wird nicht verwendet, da ROS über einen eigenen Mechanismus verfügt. Folglich wird der Parameter rawlog\_file auf einen leeren Wert gesetzt werden. Auch verfügt ROS über eine eigene Visualisierungslösung, daher wird die grafische Anzeige des MRPT Frameworks über den Parameter SHOW\_PROGRESS\_IN\_WINDOW=0 deaktiviert. Um zwischen den beiden RO-SLAM Verfahren zu wechseln, wird der Parameter insertAsMonteCarlo angepasst werden.



# 6.

---

## Evaluation

---

### 6.1. Batterielaufzeit

Beim Test der Batterielaufzeit wurden zwei UWB Modul in einem Abstand von 4,7 m aufgestellt. Beide UWB Modul hatten eine direkte LoS zu einandern. Über den kompletten Zeitraum wurden Entfernungsmessungen mit einer Rate von 10 Hz durchgeführt. Als Testprogramme wurden dabei DW1000Ranging\_ANCHOR und DW1000Ranging\_TAG aus dem GitHub Projekt [34] verwendet.

Der Anker hat nach ca. 17 h seinen Dienst eingestellt, wenig später folgte ihm der Tag. Deutlich höhere Batterielaufzeiten können dadurch erzielt werden, dass die Senderate reduziert wird und die Stromsparfunktionen sowohl des DWM1000 als auch des ATmega328/P genutzt werden.

### 6.2. Kalibrierung

Um die Antennenverzögerung pro UWB Modul zu bestimmen, werden zwei Kalibriervorgänge durchgeführt. Der erste Kalibriervorgang orientiert sich an den Herstellervorgaben aus [37]. Dazu werden drei UWB Module an die Spitzen eines gleichseitigen Dreieckes positioniert. Um das Dreieck zu konstruieren, wird in der Mitte des Dreiecks eine ausgemessene Maurerschnur befestigt, darüber wird eine Winkelschablone gelegt und dann reihum die Spitzen des Dreiecks auf dem Boden eingezeichnet, siehe Abbildung 6.1. Die Seitenlänge  $a \approx 1,73$  m wird dabei aus dem Umkreisradius  $r_u = 1$  m mit der Gleichung 2.5 berechnet.

Initial wird die Antennenverzögerung bei allen UWB Modulen auf null gesetzt und danach reihum von jedem UWB Modul eintausend Entfernungsmessungen zu den beiden benachbarten UWB Modulen durchgeführt. Insgesamt entstehen dabei sechstausend Datensätze, die mittels dem LGS- bzw. des DecaWave Kalibrierungsverfahren ausgewertet werden, siehe Tabelle 6.1

Das LGS Kalibrierungsverfahren lieferte für jeden Durchlauf reproduzierbare Ergebnisse. Da jedes UWB Modul herstellungsbedingte Unterschiede aufweist, ist es Plausibel das alle Antennenverzögerungen in einem ähnlichen Wertebereich liegen. Anders sieht es bei dem DecaWave Kalibrierungsverfahren aus. Hier ergeben sich für jeden Durchlauf andere Wertekombinationen, bedingt durch die Zufallskomponente bei der Erstellung der Kandidatenliste. Dieses Verhalten von Genetischen Algorithmen ist bekannt und muss daher bei der Auswahl

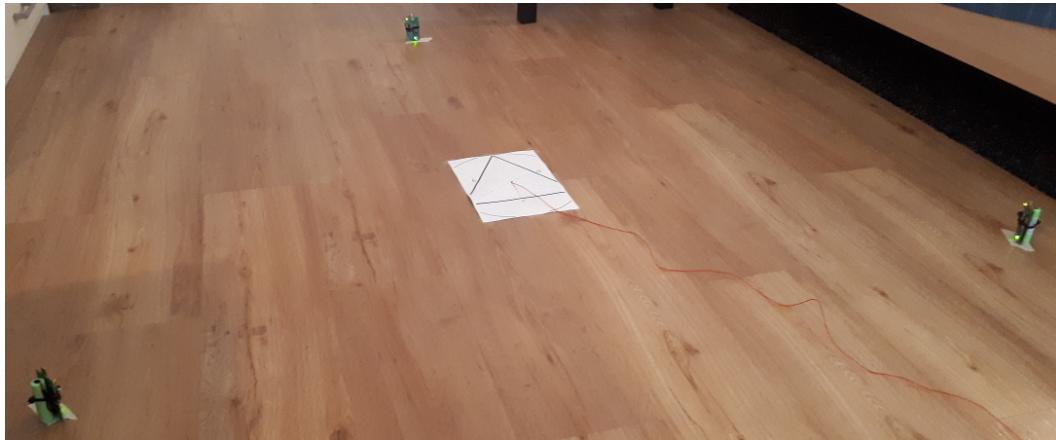


Abbildung 6.1.: Versuchsaufbau für die Kalibrierung von drei UWB Modulen.

UWB Modul	LGS [ns]	DecaWave [ns]	DecaWave 1 [ns]	DecaWave 2 [ns]
176	257,45	242,48	232,21	254,50
177	257,49	238,47	249,38	254,73
178	257,11	257,17	255,36	215,07

Tabelle 6.1.: Berechnete Werte für die Antennenverzögerung pro UWB Modul.

der Wertekombinationen berücksichtigt werden. Am Beispiel der Spalte DecaWave 1 wird es sehr deutlich. Hier beträgt der Unterschied zwischen dem größten und kleinsten Wert ca. 23 ns und entspricht damit einer Abweichung von ca. 6 m zwischen diesen beiden UWB Modulen.

Die Wertekombinationen aus den Spalten LGS und DecaWave werden abwechselnd als Antennenverzögerungen in die UWB Module eingetragen und für jede Spalte jeweils eine weitere Messreihe, wie oben beschrieben, aufgezeichnet. Die Ergebnisse sind in der Tabelle 6.2<sup>1</sup> aufgeführt.

Mit einer initialen Antennenverzögerung von null beträgt die gemessene Entfernung zwischen zwei UWB Modulen ca. 156 m bei einer tatsächlichen Entfernung von ca. 1,73 m. Eine Standardabweichung von ca. 2 cm entspricht dabei den Produkteigenschaften mit denen DecaWave wirbt.

Werden die Antennenverzögerung aus der LGS Kalibrierungsverfahren benutzt, verbleibt die Standardabweichung in der gleichen Größenordnung wie im unkalibrierten Zustand. Jedoch nähert sich jetzt die gemessenen Entfernungen der tatsächlichen sehr stark an und bildet diesen sehr gut ab. Ganz im Gegensatz zu dem DecaWave Kalibrierungsverfahren.

In der Abbildung 6.2 wurden die gemessenen Entfernungen als Histogramm dargestellt. Gut zu erkennen ist die Normalverteilung der Messwerte um den Mittelwert.

Im zweiten Kalibriervorgang werden alle fünf UWB Module eingeschlossen. Jetzt reicht ein Dreieck nicht mehr aus, statt dessen wird ein regelmäßiges Fünfeck verwendet, siehe Abbildung 6.3. Dieses lässt sich mathematisch ähnlich gut beschreiben wie das Dreieck. Aus der Seitenlänge  $a = 4,5$  m, zwischen zwei benachbarten Spitzen, wird über die Gleichung 2.7

<sup>1</sup>Bei der Standardabweichung von über einem Meter handelt es sich um einen einzelnen Messfehler in der Messreihe.

Kalibrierung	Tag	Anker	Entfernung [m]	$\bar{x}$ [m]	$\sigma$ [m]	Min [m]	Max [m]
Keine	176	177	1,732	156,108	0,018	156,050	156,170
	176	178	1,732	155,929	0,021	155,870	156,000
	177	176	1,732	156,106	1,025	156,020	188,470
	177	178	1,732	156,016	0,023	155,930	156,090
	178	176	1,732	156,067	0,022	156,010	156,130
	178	177	1,732	155,997	0,019	155,930	156,060
LGS	176	177	1,732	1,695	0,019	1,640	1,750
	176	178	1,732	1,795	0,022	1,740	1,880
	177	176	1,732	1,656	0,017	1,590	1,700
	177	178	1,732	1,751	0,023	1,670	1,810
	178	176	1,732	1,773	0,026	1,700	1,850
	178	177	1,732	1,712	0,020	1,660	1,780
DecaWave	176	177	1,732	11,883	0,020	11,820	11,950
	176	178	1,732	6,261	0,021	6,200	6,340
	177	176	1,732	11,857	0,015	11,810	11,900
	177	178	1,732	7,406	0,023	7,340	7,480
	178	176	1,732	6,182	0,025	6,100	6,270
	178	177	1,732	7,458	0,019	7,390	7,520

Tabelle 6.2.: Stochastische Eigenschaften der UWB Module ohne und mit Antennenkalibrierung bei einem Abstand von 1,73 m.

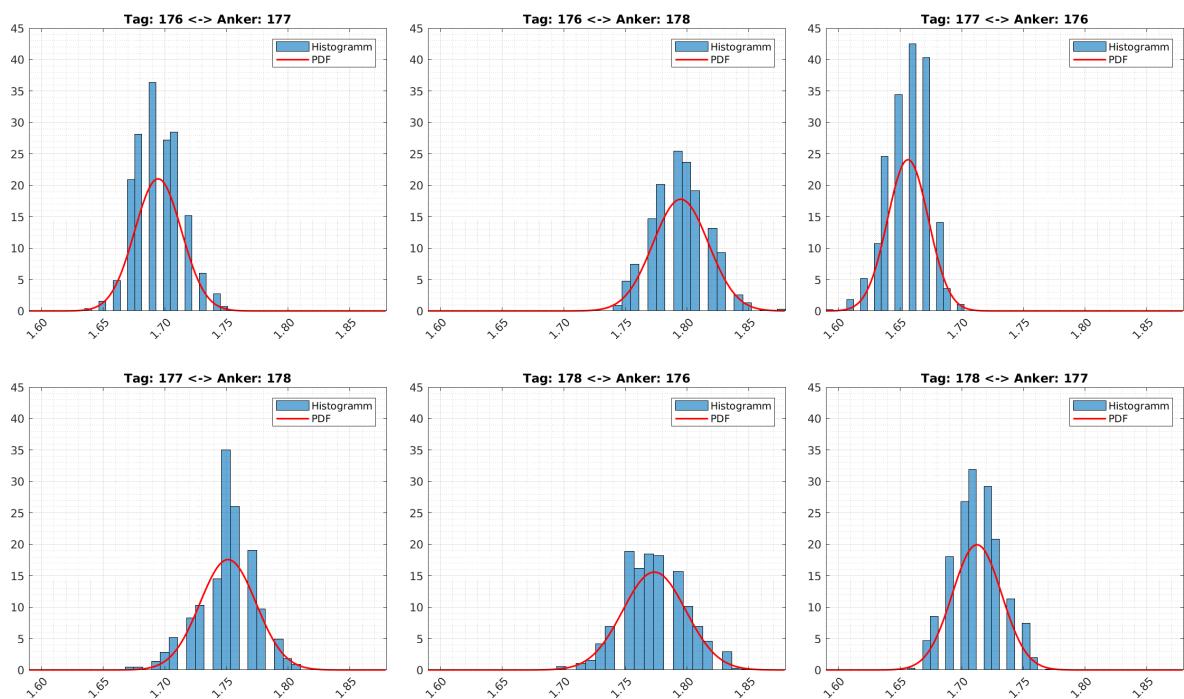


Abbildung 6.2.: Histogramm und Wahrscheinlichkeitsdichtefunktion der kalibrierten Entfernungsmessungen.



Abbildung 6.3.: Versuchsaufbau für die Kalibrierung von fünf UWB Modulen.

UWB Modul	LGS [ns]
176	257,30
177	256,67
178	256,67
179	256,41
180	257,20

Tabelle 6.3.: Berechnete Werte für die Antennenverzögerung pro für das regelmäßige Fünfeck.

die Entfernung  $d$ , zwischen den diagonalen Spitzen, berechnet. Der Umkreisradius  $r_u$  ergibt sich aus der Gleichung 2.6.

Für das regelmäßige Fünfeck wurde nur noch das LGS Kalibrierungsverfahren verwendet, die Antennenverzögerungen pro UWB Modul sind dabei in der Tabelle 6.3 aufgeführt.

### 6.3. Entfernungsmessung

Um die Charakteristik der Entfernungsmessung zu bestimmen, wird eine LoS- und drei NLoS Messreihen aufgezeichnet. Jede Messreihe beginnt bei einer Entfernung von einem Meter zwischen Tag und Anker. Pro Entfernung werden jeweils fünfhundert Messwerte aufgezeichnet. Danach wird die Entfernung um einen halben Meter erhöht, indem der Anker verschoben wird. Dies erfolgt solange bis eine Entfernung von neun Meter erreicht ist. Somit enthält jede Messreihe siebzehn Entfernung mit achttausendfünfhundert Entfernungsmessungen.

Bei der ersten NLoS Messreihe wird ein  $19 \times 12 \times 10$  cm großer, mit Wasser gefüllter Kunststoffbehälter in einem Abstand von 2 cm vor der UWB Antenne platziert, siehe Abbildung A.7b. Bei den letzten zwei NLoS Messreihen wird ein  $44 \times 25$  cm großes Aluminiumblech mit einer Dicke von 0,5 mm vor die UWB Antenne platziert, jeweils in einem Abstand von 5 cm und 50 cm, siehe Abbildung A.7c und Abbildung A.7a.

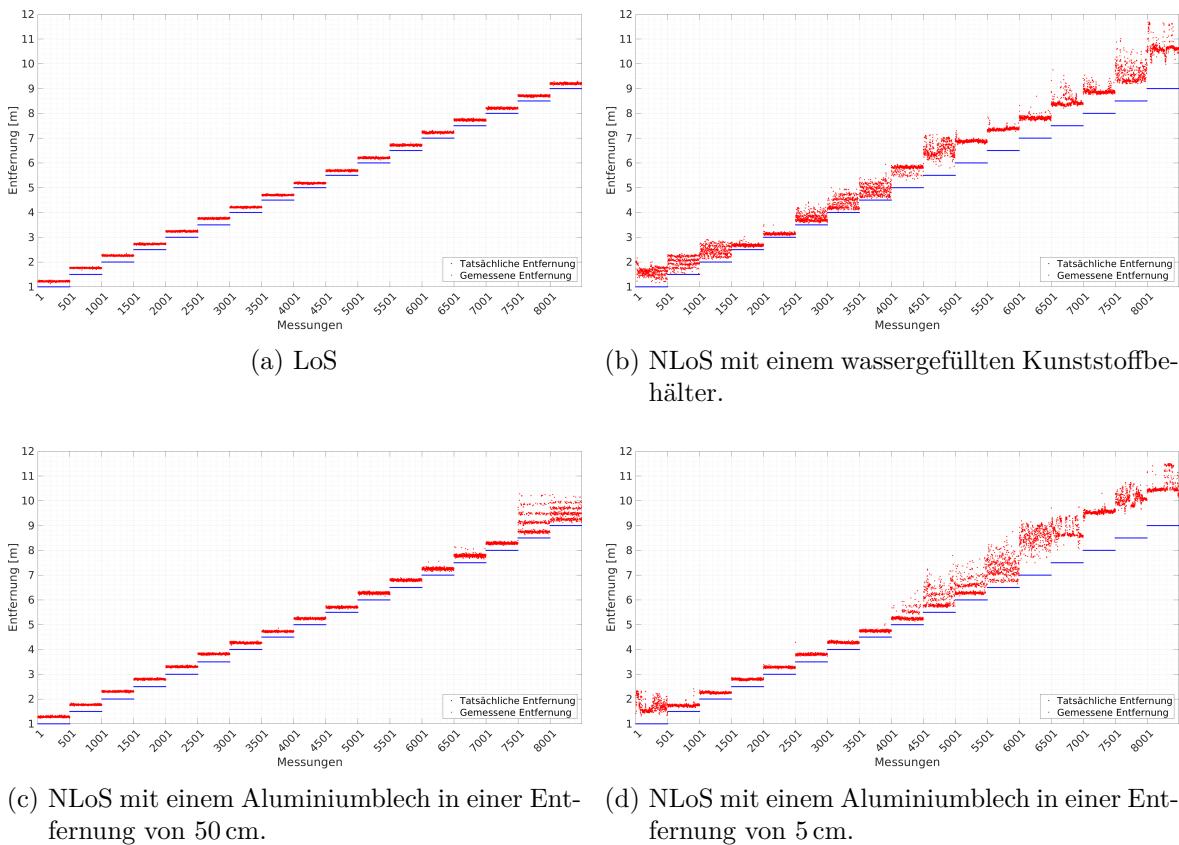


Abbildung 6.4.: Tatsächliche und gemessene LoS- und NLoS Entfernungen.

Das auffälligste Merkmal der LoS- als auch der NLoS Entfernungsmessungen ist die systematische Verschiebung aller Messwerte um ca. 20 cm deren Ursache ungeklärt bleibt, siehe Abbildung 6.4.

Bei der LoS Messreihe beträgt die Standardabweichung, wie bereits bei der Kalibrierung, ca. 2 cm. Mit steigender Entfernung steigt auch die Standardabweichung von ca. 1,8 cm bei einem Meter auf ca. 2,3 cm bei neun Meter an. Größere Ausreißer sind bei den Messwerten nicht vorhanden, siehe Abbildung 6.4a und Tabelle B.5.

In der Abbildung 6.4b ist deutlich zu erkennen, dass Wasser einen sehr störenden Einfluss auf die Entfernungsmessung ausübt. Die Messwerte streuen über einen sehr großen Bereich, das spiegelt sich auch in der Standardabweichung, die im besten Fall ca. 5 cm und im schlimmsten Fall ca. 34 cm beträgt, siehe Tabelle B.6.

Ähnlich verhält es sich bei der NLoS Messreihe mit einem Aluminiumblech, das sehr nah vor der UWB Antenne platziert wurde, siehe Abbildung 6.4d und Tabelle B.8. Einzig im Bereich von zwei bis fünf Meter beträgt die Standardabweichung ca. 2 cm.

Keinen besonders großen Einfluss hat das Aluminiumblech, wenn es sich in einer Entfernung von 50 cm befindet, siehe Tabelle B.7. Die Werte der Standardabweichung ähneln der LoS Messreihe. Nur im Bereich zwischen achteinhalb und neun Meter steigt die Standardabweichung auf einen Wert von ca. 30 cm an. Dies könnte darauf zurückzuführen sein, das im hinteren Bereich des Raums Metallstühle auf den Tischen standen, siehe Abbildung A.7a.

## 6.4. Trajektorie

Der Softwarearchitektur, siehe Abbildung 5.2, ist zu entnehmen, dass die Odometriedaten eine wichtige Rolle für den RO-SLAM spielen. Insgesamt stehen drei Odometriequellen zur Verfügung. Einmal die Inkrementalgeber der Antriebseinheit und zwei laserbasierte Odometrieeen. Alle drei Verfahren werden im Folgenden auf ihre Genauigkeit untersucht.

Für die Untersuchung werden zwei Messfahrten durchgeführt. Bei der ersten Messfahrt entspricht die Trajektorie einem abgerundeten Rechteck und bei der zweiten wird das abgerundete Rechteck um mehrere Kreisbahnen erweitert. In beiden Fällen kehrte die Roboterplattform zu der Startposition zurück um den Versatz berechnen zu können. Im Folgenden wird nur noch die zweite Messfahrt betrachtet, da bei der ersten alle Verfahren in etwa gleich abgeschnitten haben, siehe Abbildung A.8.

Alle ROS Nachrichten, der beiden Messfahrten, wurden aufgezeichnet und in separaten Bag-Dateien abgespeichert. Bevor die Bag-Dateien mit den laserbasierten Odometriequellen untersucht werden können, müssen die Odometriedaten der Inkrementalgeber entfernt werden, um widersprüchliche Informationen aus mehreren Odometriequellen zu verhindern. Für die Filterung der Bag-Dateien ist das Kommandozeilenprogramm rosbag zuständig. Mit dem Parameter filter und einem Python-Ausdruck werden alle Transformationsnachrichten zwischen dem Roboterplattform- und dem Odometrie-Koordinatensystem entfernt.

Mit dem ROS Modul hector\_trajectory\_server und dem Skript trajectory\_to\_csv.py wurden die Trajektorien der drei Odometriequellen in eine CSV-Datei gespeichert, um diese auswerten zu können. Dabei wurden aus den ungefilterten Aufzeichnungen die Trajektorie der Inkrementalgeber extrahiert. Mit den ROS Modulen für die laserbasierten Odometrieeen und den gefilterten Aufzeichnungen wurden die laserbasierten Trajektorien extrahiert.

Das ROS Modul hector\_trajectory\_server hat die Möglichkeit, die Trajektorie im Odometrie- und im Karten-Koordinatensystem aufzunehmen. Die Transformation zwischen dem Karten- und dem Roboterplattform-Koordinatensystem wird durch das ROS Modul hector\_mapping anhand der 2D-Laser-Entfernungsmessung bestimmt. Dadurch eignet es sich ideal, um als Ground Truth Trajektorie verwendet zu werden. Diese Ground Truth Trajektorie wird dann mit den Trajektorien verglichen, die im Odometrie-Koordinatensystem der jeweiligen Odometriequelle aufgezeichnet wurde.

Sowohl die Karte der Umwelt als auch die verfahrenen Trajektorien sind in der Abbildung 6.5 dargestellt. Die Ground Truth Trajektorie, mit der die anderen Trajektorien verglichen werden, ist in der Abbildung 6.5a zu sehen. Deutlich zu erkennen ist, dass die Trajektorie der Inkrementalgeber sowohl in der X- als auch in der Y-Achse gestaucht ist, siehe Abbildung 6.5b. Die laserbasierte Odometrie des LSM Verfahrens hat große Problem mit den gefahrenen Kreisbahnen und driftet zum Ende hin sehr stark von der Startposition ab, siehe Abbildung 6.5c. Einzig die laserbasierte Odometrie des RF2O Verfahrens bildet die gefahrene Trajektorie gegenüber dem Ground Turth gut ab, auch wenn die Start- und Zielpositionen voneinander abweichen, siehe Abbildung 6.5d.

Bei der stochastischen Auswertung war eine direkte zeitliche Zuordnung der Messwerte aller drei Verfahren zum Ground Truth nicht möglich, da jede Odometriequelle eine spezifische Verarbeitungsdauer besitzt. Deshalb wird pro Sekunde der Mittelwert der verfügbaren Po-

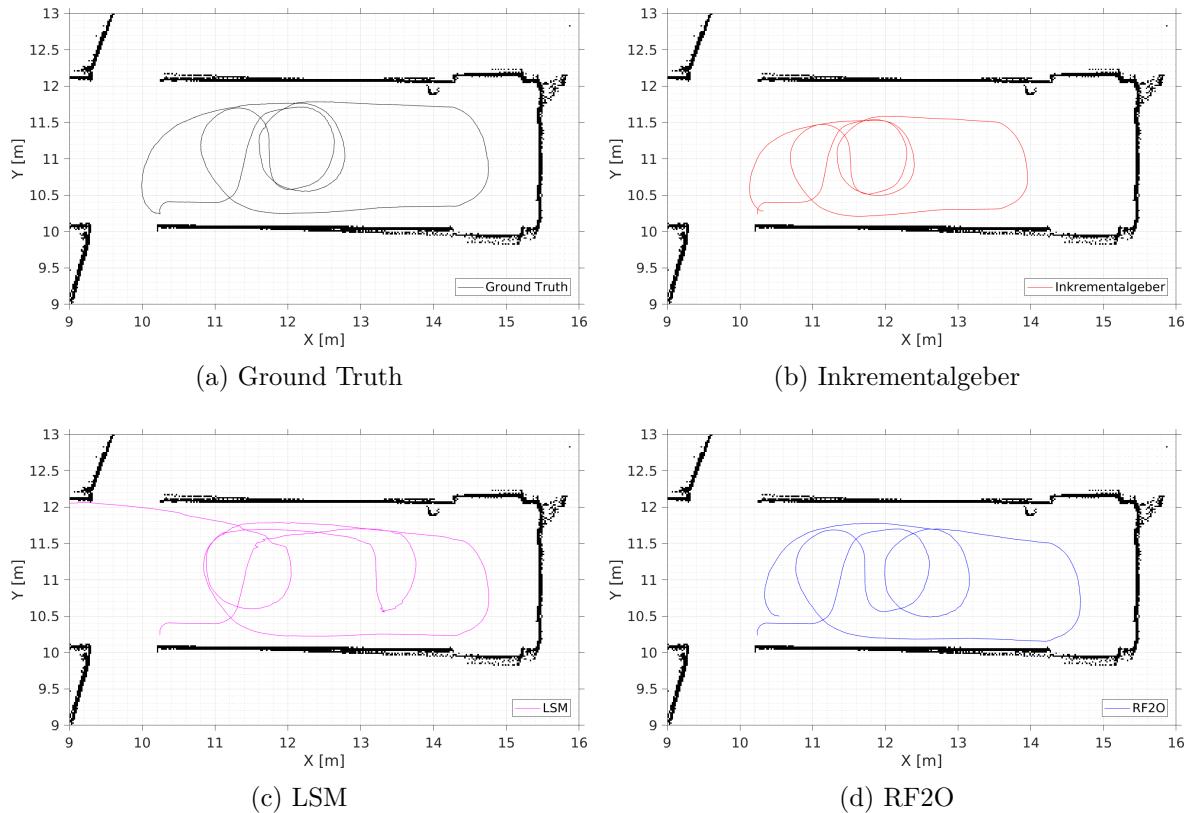


Abbildung 6.5.: Die Trajektorien der verschiedenen Odometriequellen.

Odometriequelle	$\bar{x}$ [m]	$\sigma$ [m]	$SE_{\bar{x}}$ [m]	Min [m]	Max [m]
Inkrementalgeber	0,327	0,203	0,017	0,000	0,806
LSM	0,875	0,929	0,079	0,000	2,477
RF2O	0,247	0,180	0,015	0,002	0,558

Tabelle 6.4.: Stochastische Eigenschaften der Entferungen zwischen der Odometrie- und der Ground Truth Position.

sitionen gebildet und aus dieser dann die Entfernung zur Ground Truth Position berechnet, siehe Tabelle 6.4.

Bestätigt wird die visuelle Einschätzung durch die stochastischen Auswertungen der Entfernung der Position zur Ground Truth Position, siehe Tabelle 6.4. Das LSM Verfahren weicht im Mittelwert und bei der Standardabweichung deutlich von den Inkrementalgebern und dem RF2O Verfahren ab. Das der Mittelwert und die Standardabweichung zwischen den Inkrementalgebern und dem RF2O Verfahren nur sehr geringe Unterschiede aufweisen, jedoch große Unterschiede in der visuellen Trajektorie festzustellen sind, kann nur damit erklärt werden, dass sich die aufsummierten Fehler der Inkrementalgebern bei der Rückfahrt wieder aufheben.

Da die Trajektorie des laserbasierten RF2O Verfahren am besten der Ground Truth Trajektorie entspricht, wird diese als Odometriequelle für das RO-SLAM Verfahren verwendet.

## 6.5. RO-SLAM

### 6.5.1. Positionsschätzung

Im vorherigen Abschnitt wurde bereits die Trajektorie der verschiedenen Odometriequellen untersucht. Die Entscheidung fiel auf das RF2O Verfahren, mit dem nun die Positionsschätzung des RO-SLAM Verfahren untersucht wird. Es wird dabei die Trajektorie mit dem abgerundeten Rechteck mit mehrere Kreisbahnen verwendet, siehe Abbildung 6.5. Da auch die Entfernungsmessungen der UWB Module einen Einfluss auf die Positionsschätzung haben, werden neben den realen auch virtuelle UWB Modulen verwendet. Die virtuellen UWB Module zeichnen sich dadurch aus, dass ihre Entfernungsmessungen keine Fehler aufweisen und daher als perfekt angesehen werden können.

In der Abbildung 6.6 ist die tatsächliche Trajektorie und die des RO-SLAM Partikel Filters abgebildet. Die oberen zwei Abbildungen verwenden die Entfernungsmessungen der realen UWB Module und die unteren zwei Abbildungen die Entfernungsmessung von virtuellen UWB Module. Auf der linken Seite befinden sich die Positionsschätzungen die die Ground Truth Trajektorie gut abbilden und auf der rechten Seite die die sie schlecht abbilden. Während bei den realen UWB Modulen die Positionsschätzung sehr große Abweichungen von der Ground Truth Trajektorie besitzt, bildet die Positionsschätzung der virtuellen UWB Module die Ground Truth Trajektorie deutlich besser ab, selbst bei der schlechten Abbildung weist die Positionsschätzung nur einen leichten Winkelfehler auf.

Das führt zu der Vermutung, dass die Qualität der Entfernungsmessung einen entscheidenden Einfluss auf die Positionsschätzung des RO-SLAM Verfahrens besitzt. Je geringe die Abweichungen der Entfernungsmessung von der tatsächlichen Entfernung desto besser. Bemerkenswert ist es auch, dass die Reproduzierbarkeit der Ergebnisse mit den virtuellen UWB Modulen deutlich eher gegeben war, als mit den realen UWB Modulen.

### 6.5.2. Positionsschätzung der UWB Module

Mit den gleichen Datensätzen wie in Abbildung 6.6 wurden nun in der Abbildung 6.7 die Fehlerellipsen für die UWB Module eingezeichnet. Um die Fehlerellipsen darstellen zu können, wurde diese mit einem Faktor von 10 skaliert. Die Ground Truth Position der UWB Modul ist an dem schwarzen Kreuz erkennbar. Die Roboterplattform wird durch einen blauen Stern symbolisiert, der sich zu unterschiedlichen Zeitpunkten auf der Trajektorie befindet. Die gepunkteten Linien ordnen der Roboterplattform, zu einem bestimmten Zeitpunkt, die Positionsschätzung der UWB Module zu.

Sowohl mit den realen als auch mit den virtuellen UWB Modulen verringert sich die Unsicherheit der Positionsschätzung der UWB Module kontinuierlich über die Zeit. Durch die ungenaue Positionsschätzung der Roboterplattform durch die realen UWB Module sind auch die Positionsschätzungen der UWB Modul entsprechend verschoben im Bezug zu ihren Ground Truth Position, siehe Abbildung 6.7a und Abbildung 6.7b. Besser verhält es sich bei den virtuellen UWB Modulen, hier entspricht die Positionsschätzung der UWB Module dem der Ground Truth Position, siehe Abbildung 6.7c. Der Winkelfehler in der Positionsschätzung

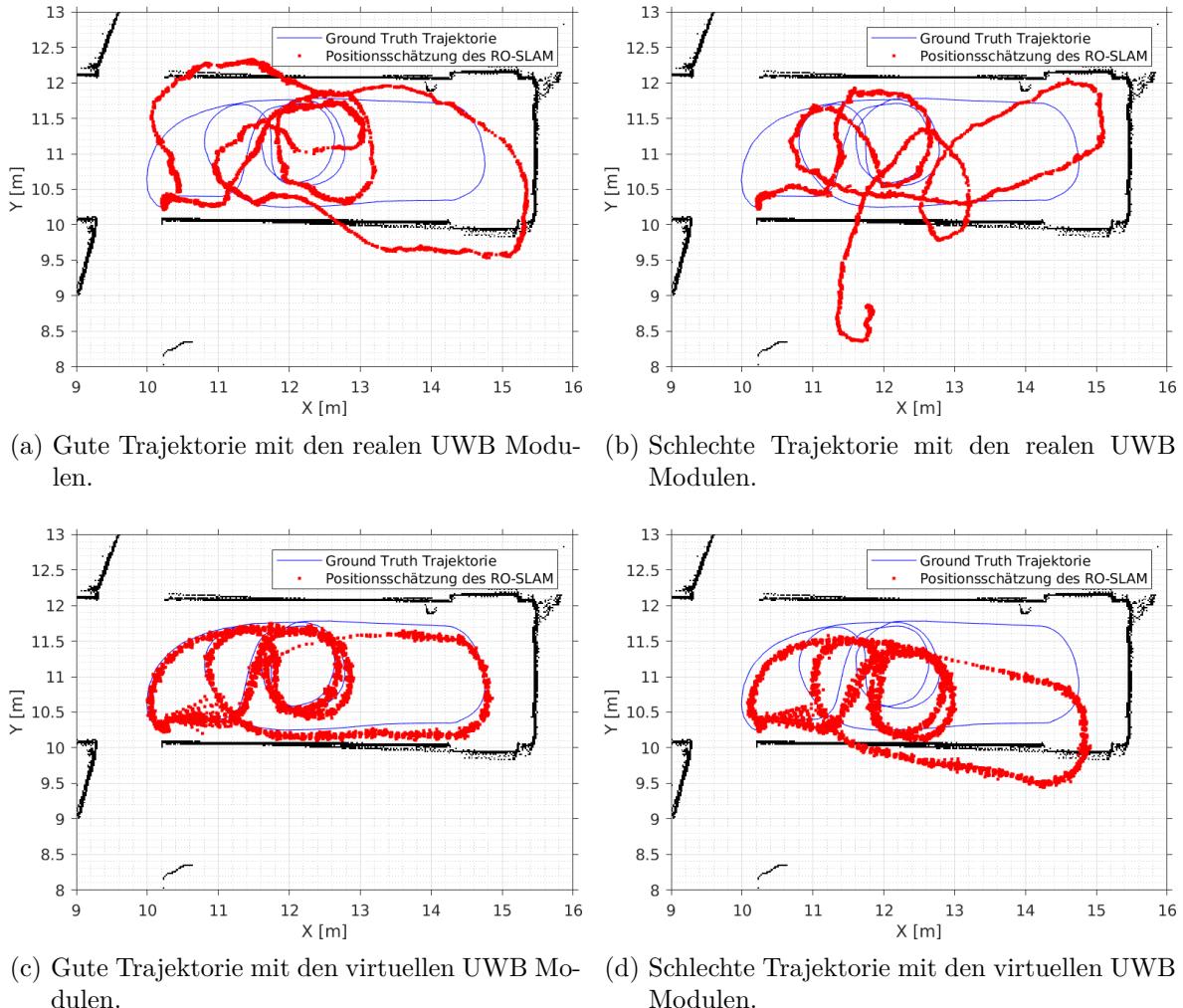


Abbildung 6.6.: Resultierende Trajektorie aus der Positionsschätzung des RO-SLAM mit realen und virtuellen UWB Modulen.

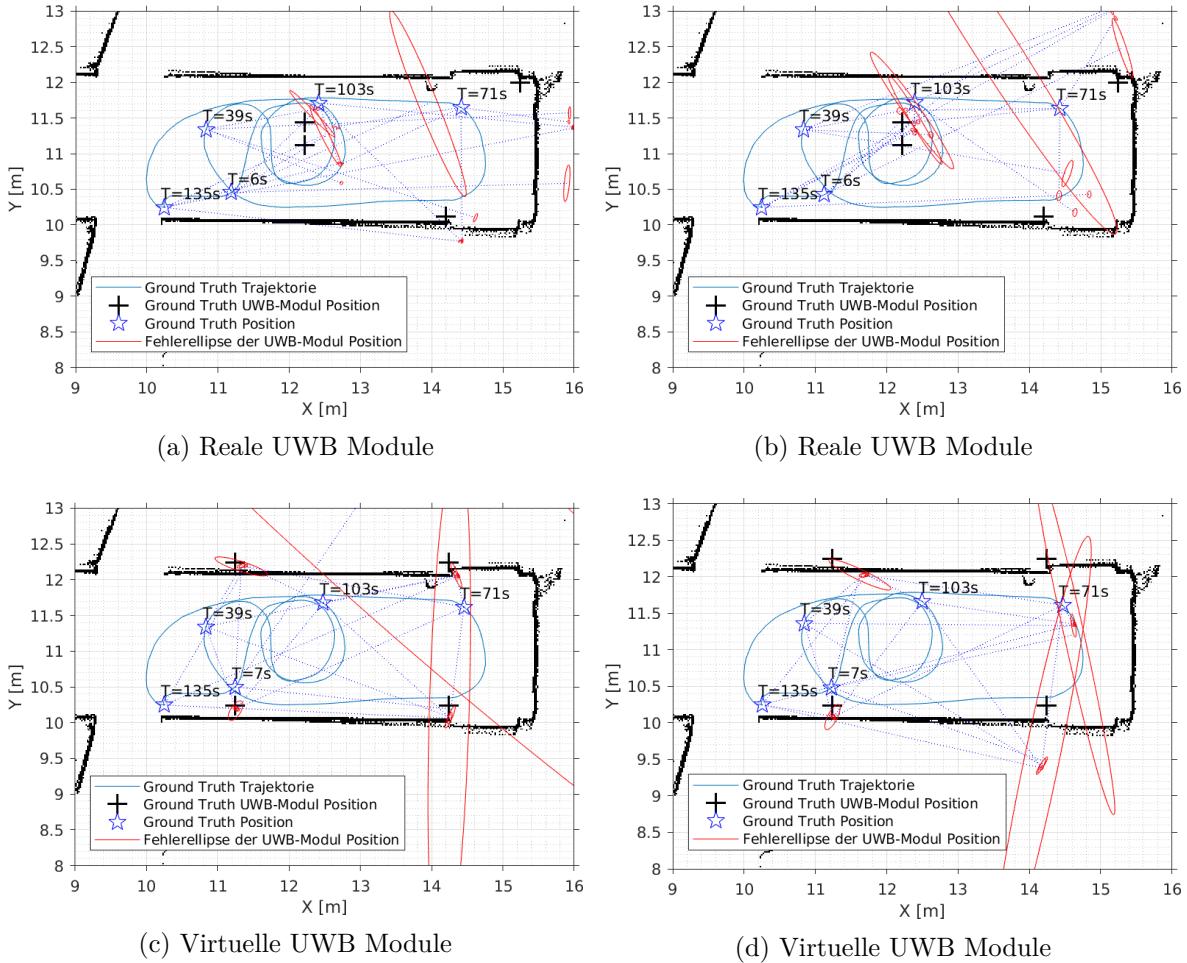


Abbildung 6.7.: Positionsschätzungen der realen und virtuellen UWB Modulen zu bestimmten Zeitpunkten.

aus der Abbildung 6.6d spiegelt sich ebenfalls in der Positionsschätzung der UWB Module wieder, siehe Abbildung 6.7.

Anzumerken ist, dass das RO-SLAM Verfahren von der Annahme ausgeht das die Position der UWB Module sich über die Zeit nicht ändert. Während der Auswertung der Daten konnte jedoch mehrfach beobachtet werden, dass sich die Positionen der UWB Module über die Zeit änderten. Somit ist nicht ausgeschlossen, dass die Positionsänderungen der UWB Module verfolgt werden kann, sobald die WDF durch das EKF Verfahren repräsentiert wird.

In den Tabelle 6.5 und Tabelle 6.6 sind die Entferungen der Positionsschätzung der UWB Module zu den Ground Truth Positionen an ausgewählten Zeitpunkten aufgelistet. Wie bereits zu erwarten schneiden die virtuellen UWB Module deutlich besser ab als ihre realen Vertreter. Neben einer Verbesserung der Positionsschätzung der UWB Module ist eine Verschlechterung ebenfalls möglich, siehe linke Spalte 179 und rechte Spalte 4. Die Verschlechterung tritt sowohl die virtuellen als auch die realen UWB Module gleichermaßen.

Einige Entfernungen kommen pro Spalte doppelt bzw. dreifach hintereinander vor. Diese entstehen dadurch, das das MRPT ROS Modul in großen Zeiträumen keine neuen Positionsschätzungen für diese UWB Module bereitgestellt hatte. Aus diesem Grund wurde die letzte

	177	178	179	180	177	178	179	180
6 s	0,317 m	0,159 m	1,561 m	1,601 m	0,396 m	0,212 m	0,488 m	2,439 m
39 s	0,739 m	0,261 m	0,816 m	0,414 m	0,426 m	0,162 m	1,049 m	0,687 m
71 s	0,584 m	0,404 m	0,883 m	0,406 m	0,396 m	0,175 m	0,905 m	0,360 m
103 s	0,536 m	0,460 m	0,972 m	0,424 m	0,240 m	0,167 m	1,046 m	0,441 m
135 s	0,567 m	0,490 m	0,975 m	0,396 m	0,281 m	0,170 m	1,036 m	0,709 m

Tabelle 6.5.: Entfernung der Positionsschätzung der realen UWB Module zu den Ground Truth Positionen.

	1	2	3	4	1	2	3	4
7 s	0,074 m	0,049 m	1,572 m	0,102 m	0,161 m	0,756 m	1,106 m	0,438 m
39 s	0,049 m	0,165 m	0,205 m	0,100 m	0,156 m	0,828 m	0,986 m	0,486 m
71 s	0,050 m	0,177 m	0,205 m	0,108 m	0,162 m	0,829 m	0,956 m	0,491 m
103 s	0,050 m	0,177 m	0,226 m	0,158 m	0,153 m	0,861 m	0,976 m	0,528 m
135 s	0,051 m	0,177 m	0,228 m	0,159 m	0,154 m	0,861 m	0,976 m	0,529 m

Tabelle 6.6.: Entfernung der Positionsschätzung der virtuellen UWB Module zu den Ground Truth Positionen.

bekannte Entfernung angegeben. Die Ursache für dieses Phänomen konnte bis zum Schluss nicht geklärt werden.

### 6.5.3. Konvergenz der WDF

Solange die Positionsschätzung der UWB Modul Mehrdeutigkeiten aufweist, werden als WDF das SOG- oder das MC Verfahren eingesetzt. Liegen keine Mehrdeutigkeiten mehr vor, wird die WDF mit dem EKF Verfahren abgebildet. Das bedeutet, dass jedes UWB Modul nur noch über die Parameter Mittelwert und Kovarianz beschrieben wird, was zu einer Reduktion der genutzten Ressourcen der Verarbeitungseinheit führt.

In der Tabelle 6.7 sind die Zeitspannen und die zurückgelegte Entfernung der Roboterplattform pro UWB Modul aufgelistet, bis die WDF durch das EKF Verfahren abgebildet wird. Sowohl mit den virtuellen als auch mit den realen UWB Modul findet die Umwandlung bereits nach weniger als 1 m bzw. weniger als 10 s statt. Einzig bei den virtuellen UWB Modulen benötigte das SOG Verfahren deutlich länger. Im Allgemeinen konvergieren sowohl das SOG- als auch das MC Verfahren gleich schnell.

UWB Modul	ID	Verfahren	Zeitspanne [s]	Entfernung [m]
Real	177	SOG	3,8	0,505
Real	178	SOG	4,4	0,616
Real	179	SOG	5,0	0,708
Real	180	SOG	7,0	1,047
Real	177	MC	5,6	0,821
Real	178	MC	8,3	1,207
Real	179	MC	4,8	0,670
Real	180	MC	8,0	1,168
Virtuell	1	SOG	3,0	0,357
Virtuell	2	SOG	10,0	1,396
Virtuell	3	SOG	15,5	2,245
Virtuell	4	SOG	4,5	0,606
Virtuell	1	MC	1,0	0,047
Virtuell	2	MC	6,1	0,868
Virtuell	3	MC	1,0	0,047
Virtuell	4	MC	5,5	0,761

Tabelle 6.7.: Die Zeitspanne und die zurückgelegte Entfernung der Roboterplattform bis die WDF durch das EKF Verfahren abgebildet wird.

# 7.

---

## Zusammenfassung und Ausblick

---

Im Grundlagenkapitel wurden zu erste der Unterschied zwischen der Entfernungsmessung mittels Triangulation und der Trilateration beschrieben. Die Triangulation bestimmt die Entfernung durch das Messen der Winkel zwischen mehreren Referenzpunkten, während die Trilateration die Entferungen anhand der Signallaufzeit bestimmt. Die Trilateration wird von den UWB Modulen verwendet um Nachrichten auszutauschen. Durch den Nachrichtenaustausch ist es auch möglich die Entfernung zwischen zwei UWB Modulen zu bestimmen. Dazu wird der Nachrichtenversand zu einem zukünftigen Zeitpunkt geplant, um den Zeitstempel des Sendevorgangs in die Nachricht einzubetten. Das empfangende UWB Modul ist nun im Besitz aller Informationen um die Entfernung zu errechnen. Dies ist unter dem Namen ST-TWR Verfahren bekannt. Eine Verbesserung stellt das DS-TWR Verfahren dar, das für die Entfernungsmessung verwendet wird, da es den Fehler der lokalen Zeitgeber minimiert.

Im Geometrieabschnitt wurden die mathematischen Gleichungen für das Konstruieren und Berechnen der relevanten Längen des gleichseitigen Dreiecks und eines regelmäßigen Fünfecks beschrieben.

Die Wahrscheinlichkeitstheorie legt den Grundstein um die Funktionsweise der verschiedenen SLAM Varianten zu verstehen. Dabei wurden die Konzepte der Zufallsvariablen, der einfachen und mehrdimensionalen Normalverteilung und deren Gesetzmäßigkeiten wie die bedingte Wahrscheinlichkeit, die Abhängigkeiten zwischen Zufallsvariablen und der Satz von Bayes vorgestellt.

Mit einem Zustandschätzer ist es möglich, Abschätzungen über den zukünftigen Zustand eines Systems zu erstellen. Der Zustand beschreibt dabei alle Aspekte eines Roboters und seiner Umwelt die einen Einfluss auf die Zukunft haben können. Die Abschätzungen werden dabei durch die Steuerbefehle, die der Roboter an die Aktorik sendet, und die Wahrnehmungen der Umwelt gesteuert. Das interne Wissen des Roboters über den Zustand seiner Umwelt wird dabei als Belief bezeichnet. Die Basis aller Zustandschätzer bildet dabei der rekursive Bayes Filter, der jeden Zustand in zwei Schritten schätzt. Im Prognose-Schritt wird der nächste Zustand mit den Steuerbefehlen vorhergesagt und dann im Korrektur-Schritt mit den Wahrnehmungen korrigiert. Bei dem Bayes Filter handelt es sich um einen sehr abstrakten Algorithmus, der durch den Kalman Filter konkret umgesetzt wird. Der Kalman Filter nutzt dabei eine mehrdimensionale Normalverteilung um den Belief zu repräsentieren. Dadurch entstehen bei der Nutzung von nicht linearen Funktionen, die bei Rotationsbewegungen auf-

treten, aber auch Probleme. Diese werden durch den EKF mittels einer Linearisierung der nicht linearen Funktion gelöst.

Bedingt durch die Verwendung einer Normalverteilung, können sowohl der Kalman Filter als auch der EKF keine multimodalen Verteilungen darstellen. Diese Einschränkungen können durch den Partikel Filter umgangen werden. Dieser nutzt eine Menge von Partikeln um den Belief darzustellen. Dadurch ist es möglich eine beliebige Verteilung abzubilden. Jedes Partikel wird mit einem Gewicht versehen, um die Relevanz zu beschreiben. Im Resampling-Schritt werden die Partikel aussortiert, bei denen die Gewichtung unter einen festgelegten Grenzwert fällt, und durch neue Partikel ersetzt.

Wenn es notwendig wird, das nicht nur der Zustand des Roboters geschätzt, sondern auch gleichzeitig eine Karte der Umwelt erstellt wird, spricht man von einem SLAM Verfahren. Der EKF-SLAM gehört dabei zu den Standardverfahren der Robotik. Hierbei wird der Zustand des Roboters, als auch der der Umwelt mittels eines kombinierten Zustandsvektors modelliert. Das SLAM Verfahren kann auch mithilfe eines Partikel Filter gelöst werden, dazu muss der Zustandsvektor aufgespalten werden, da Partikel Filter in großen Zustandsräumen nicht praktikabel sind. Der Partikel Filter stellt somit durch seine Partikel eine Hypothese des Pfades dar und jedem Partikel wird dann eine Menge von Landmarken zugewiesen.

Das Grundlagenkapitel schließt mit einer kurzen Betrachtung der ROS Begrifflichkeiten ab. Hierzu gehört der ROS Master, der die Verbindung zwischen verschiedenen Verarbeitungseinheiten, auch als Knoten bezeichnet, bereitstellt. Damit eine Kommunikation zwischen den Knoten stattfinden kann, werden Nachrichten über Datenbusse übertragen.

In den ersten beiden Veröffentlichungen im Kapitel Stand der Forschung und Technik werden die Positionen der Beacons zuerst solange beobachtet, bis die Unsicherheit so gering ist, dass diese problemlos mit einem EKF-SLAM verarbeitet werden können. In der ersten Veröffentlichung werden die ungefähren Beacon-Positionen durch die Kombination von Probability Grid angenähert. Anders geht die zweite Veröffentlichung vor, hier werden die beobachteten Entfernung in einem Gitter eingetragen. Sobald sich ein Gipfel gebildet hat, wird die Positionsschätzung an den EKF-SLAM übergeben.

Die nächsten zwei Veröffentlichungen nutzen den Partikel Filter SLAM. Beim ersten wird für jedes Partikel ein Hilfspartikel Filter eingesetzt, um die radiale Verteilung zu modellieren. Der Zweite verwendet anstatt einem Hilfspartikel Filter eine Menge von Normalverteilungen die ebenfalls radial angeordnet und mit Gewichten versehen sind. Sobald beide Verfahren gegen eine Beacon-Positionen konvergiert sind, wird der Hilfspartikel Filter und die Menge von Normalverteilungen durch einen EKF ersetzt.

Einen vollkommen anderen Weg beschreibt die letzte Veröffentlichung, hier werden die Beacon-Positionen nicht in Kartesischen Koordinaten, sondern in Polarkoordinaten beschrieben.

Das Ultra-Wideband Kapitel beginnt mit der Beschreibung der Unterschiede zwischen der Übertragung von Informationen durch das Aufmodulieren auf eine sinusförmige Trägerfrequenz und der Übertragung von Informationen im Basisband durch das Erzeugen von kurzen Impulsen im Nanosekundenbereich. Die UWB Technologie verwendet das letzte Verfahren und kann dank der hohen Bandbreite auch entsprechend hohe Datenmengen übertragen.

---

Danach geht es an die Erstellung der Hardware, sprich der UWB Module. Zu den Hauptanforderungen gehört die gemeinsame Hardwareplattform, eine separate Stromversorgung und eine optionale Kommunikationsschnittstelle. Das Herzstück der UWB Module ist dabei der UWB Transceiver von DecaWave. Dieser IC sorgt für die komplette Verarbeitung und Auswertung der UWB Signale. Gesteuert wird dieser über die SPI Schnittstelle durch einen Arduino kompatiblen Mikrocontroller, dem Pro Trinket. Da der Mikrocontroller über keine eigene Kommunikationsschnittstelle zur Verarbeitungseinheit verfügt, wird diese über eine separate Kommunikationsschnittstelle gelöst. Im Vorfeld wurden zwei Prototypen der UWB Modul aufgebaut. Zum einen um die Beschaltung der elektrischen Komponenten zu testen und zum anderen um den Nachrichtenaustausch mit gleichzeitiger Entfernungsmessung auszuprobieren. Nach der erfolgreichen Prototypenphase wurde ein Platinendesign erstellt und durch einen entsprechenden Dienstleister gefertigt.

Als Software für die Steuerung des UWB Transceiver durch den Mikrocontroller wurde ein GitHub Projekt verwendet. Dieses stellte die Basis für die Kommunikation mit dem UWB Transceiver bereit. Zusätzlich waren die Kommunikationsprotokolle für die Entfernungsmessung bereits implementiert. Lediglich die Kommunikationsschnittstelle zwischen dem Mikrocontroller und der Verarbeitungseinheit, und die Integration der Entfernungsmessungen in das ROS System mussten entwickelt werden.

Nach dem die UWB Module erstellt und die Kommunikationsschnittstelle bereitstand, musste die Antennenverzögerung durch eine Kalibrierung der UWB Module bestimmt werden. Hierfür wurde zum einen das herstellerspezifische Verfahren implementiert, das auf Basis eines genetischen Algorithmus die Antennenverzögerung bestimmte. Durch die Erkenntnis, dass das Verfahren nur unzureichende Ergebnisse lieferte, wurde ein weiteres Verfahren auf der Basis eines linearen Gleichungssystems entwickelt.

Um das RO-SLAM Verfahren auszuprobieren wurde eine Roboterplattform benötigt. Die Wahl fiel auf den Robotino 2 von Festo Didactic. Diese ist mit einem holonomen Antrieb aufgerüstet und kann die Eigenbewegung über Inkrementalgeber bestimmen. Als zusätzlichen Sensor verfügt die Roboterplattform über einen 2D-Laser-Entfernungsmesser. Um die Daten zu verarbeiten und das RO-SLAM Verfahren auszuführen verfügt die Roboterplattform über eine eigene Verarbeitungseinheit.

Der Softwarearchitektur ist zu entnehmen, dass das RO-SLAM Verfahren zwei Datenquellen benötigt. Zum einen den Transformationsbaum, der die statischen Transformationen zwischen dem Mittelpunkt der Roboterplattform und dem UWB Modul bereithält, als auch die dynamischen Transformationen zwischen dem Koordinatensystem der Roboterplattform und dem Koordinatensystem der Odometrie. Zum anderen die Daten der Entfernungsmessung. Allgemein gilt, dass die Odometriedaten der Inkrementalgeber eines holonomen Antrieb durch den Schlupf der Räder verfälscht sind. Daher werden zwei laserbasierte Verfahren als alternative Odometriequelle erprobt.

Die ROS Module wurden dabei in zwei Kategorien unterteilt, Haupt- und Hilfsmodule. Zu den Hauptmodulen gehören alle Module die für die Steuerung der Roboterplattform, die UWB Entfernungsmessung und Koordinatentransformation zuständig sind und somit die Ausführung des RO-SLAM Verfahrens ermöglichen. Die Hilfsmodule stellen Funktionalitäten

bereit die im Nachgang für die Auswertung benötigt werden, dazu gehört der 2D-Laser-Entfernungsmesser, die laserbasierte Odometrie und die Belegtheitskarten.

Das RO-SLAM Verfahren wird durch das MRPT Framework bereitgestellt. Hierbei handelt es sich um eine Bibliothek die Datenstrukturen und Algorithmen aus dem aktiven Forschungsbereich der Robotik bereitstellt. Zusätzlich zu der Bibliothek werden auch fertige ROS Module bereitgestellt. Diese beinhalten sowohl den RO-SLAM auf Basis eines Hilfspartikel Filter, als auch den mit der Menge von radial angeordneten Normalverteilungen.

Das Kapitel Evaluation bezieht sich zuerst auf die Auswertung der Tauglichkeit der erstellten UWB Module und in dem letzten Abschnitt findet eine Auswertung des RO-SLAM Verfahrens statt. Es gibt zwei Arten von UWB Modulen, der Tag wird auf der Roboterplattform befestigt und von dieser mit Energie versorgt, die Anker werden auf der Messstrecke verteilt und erhalten ihre Energie aus einem Lithium-Ion Akku. Daher findet die erste Auswertung im Rahmen der Bewertung der Laufzeit statt.

Danach wird die Kalibrierung der Antennenverzögerung durchgeführt. Dazu werden die UWB Module an die Spitzen eines gleichseitigen Dreiecks positioniert und von jedem UWB Modul die Entfernung zu den beiden anderen UWB Modulen gemessen. Die Daten wurden danach von den beiden Kalibrierungsverfahren ausgewertet, die neuen Antennenverzögerungen in die UWB Module einprogrammiert und die Resultate miteinander verglichen. Danach wurden alle UWB Module an die Spitzen eines regelmäßigen Fünfecks positioniert und erneut die Entfernung zu den vier anderen UWB Modulen gemessen. Somit konnten die Antennenverzögerungen für alle fünf UWB Module bestimmt werden.

Nach der Kalibrierung wurden Entfernungsmessung mit LoS und NLoS durchgeführt. Hierfür wurde die Entfernung zwischen den UWB Modulen sukzessive von 1 m auf 9 m erhöht. Die NLoS Versuche wurden mit einem wassergefüllten Behälter und einem Aluminiumblech als Hindernis durchgeführt.

Da die Inkrementalgeber als Odometriequelle verfälschte Informationen liefern, wurden die 2D-Laser-Entfernungsmessungen aufgezeichnet und mit den laserbasierten Odometriequellen verarbeitet. Die Trajektorien aller drei Verfahren wurde im Nachgang mit einer Ground Truth Trajektorie verglichen.

Nachdem die beste Odometriequelle bestimmt wurde, fand die Untersuchung für die Positionsschätzung der Roboterplattform und der UWB Module durch das RO-SLAM Verfahren statt. Hierfür wurden sowohl reale als auch virtuelle UWB Module verwendet. Es wurde dabei die gleiche Trajektorie verfahren wie in dem vorherigen Versuch. Der Abschluss der Untersuchung des RO-SLAM Verfahrens bildete die Bestimmung der Konvergenzdauer des MC- bzw. SOG Verfahrens hin zu einem EKF.

## 7.1. Ausblick

Für zukünftige Untersuchungen besteht ein großes Potential in dem Bereich der UWB Module. Um eine längere Laufzeit zu erreichen ließe sich der Energieverbrauch durch verschiedene Maßnahmen deutlich reduzieren. Dazu zählt die Verwendung von oberflächenmontierten Baulementen (engl. Surface-Mount Device (SMD)), der verzicht auf die Ansteuerung der Statusleuchtdioden, einen schnelleren Prozessor der die volle Geschwindigkeit der SPI Schnittstelle

ausnutzen kann um sich und den UWB Transceiver nach dem Datenaustausch wieder in einen Energiesparmodus zu versetzen und einen Prozessor der über deutlich mehr Speicher verfügt, um sowohl die ROS Bibliothek verwenden zu können, als auch die Anzahl der Unterstützten UWB Modul zu erhöhen. Als Komfortfunktion wäre es sinnvoll, wenn die Identifikationsnummern der UWB Modul nicht per Software, sondern über Hardwareschalter geändert werden könnten.

Grundsätzlich sollte aber vorher geklärt werden, in wie weit sich diese Investitionen lohnen. Der Hersteller der UWB Transceiver stellt eigene UWB Module her, die wahrscheinlich bereits alle verfügbaren Optimierungen nutzen um eine möglichst genaue Entfernungsmessung zu erlangen. Hier wäre ein Vergleich der Eigenschaften beider UWB Modul interessant.

Im Bereich der RO-SLAM Verfahren wäre es eine sinnvolle Untersuchung, in wie weit sich das Ergebnis, durch die Verwendung des Polarkoordinatensystems für die Positionsbestimmung der UWB Module, verbessert.

## 7.2. Fazit

Der Aufbau der UWB Module verlief, dank der guten Spezifikation des Herstellers und den vielen Erfahrungsberichten aus diversen Internetforen, problemlos. Bereits mit den ersten beiden Prototypen konnte ein Nachrichtenaustausch und eine Entfernungsmessung durchgeführt werden. Der Datenaustausch des UWB Modul mit der Verarbeitungseinheit gestaltet sich schwieriger als erwartet. Der verfügbare Speicherplatz des ATmega328/P ließ die gleichzeitige Verwendung der Bibliothek für die Ansteuerung des UWB Transceivers und der Kommunikationsbibliothek von ROS nicht zu. Daher wurde eine Kommunikationsschnittstelle erstellt, die die Entfernungsdaten ins JSON Format kodiert und über die serielle Schnittstelle an die Verarbeitungseinheit versendet.

Mit der Standardeinstellung für die Antennenverzögerung lieferte die Entfernungsmessung, zwischen zwei UWB Modulen, deutlich höhere Entfernungswerte zurück, als diese in der Realität gewesen sind. Daher führte kein Weg an einer Kalibrierung vorbei. Dabei wurde das durch den Hersteller empfohlene, jedoch unzureichend beschriebene, Verfahren implementiert. Der Kern des Verfahrens wird durch einen Genetischen Algorithmus umgesetzt. Dies führte dazu, dass mit jeder Ausführung des Verfahrens komplett unterschiedliche Ergebnisse generiert wurden, die auch nicht plausibel erschienen. Aus diesem Grund wurde ein alternatives Verfahren auf Basis eines linearen Gleichungssystems implementiert, welches mit jedem Durchlauf reproduzierbare und plausible Ergebnisse lieferte.

Nach der Kalibrierung wurden die Entfernungswerte sowohl mit einer LoS als auch mit einer NLoS gemessen. Im LoS Fall betrug die Standardabweichung im Mittel 0,02 m. Dieser Wert entspricht vollkommen den Herstellerangaben. Im NLoS Fall stieg die Standardabweichung im Mittel auf 0,15 m an. Jedoch sollte berücksichtigt werden, dass Materialien verwendet wurden, die eine starke abschirmende bzw. hemmende Wirkung auf die UWB Signale besitzen. Erstaunlich war jedoch, dass trotz der Kalibrierung weiterhin ein systematischer Fehler in den Ergebnissen vorlag. Die Ursachen können vielfältiger Natur sein, zum einen wertet die verwendete Steuersoftware, für die UWB Transceiver, die Umgebungstemperatur nicht aus, um gegebenenfalls Korrekturen an der Entfernung durchzuführen, die laut Herstellerangaben die

Entfernungsmessung jedoch beeinträchtigen können. Zum anderen stellt der Hersteller für diese Hardwareplattform keine eigene Steuersoftware bereit, die diese Optimierungen durchführt. Daher ist man auf die Version einer freien Software angewiesen, die nur eine Kommunikationschnittstelle zu dem UWB Transceiver anbietet aber keine Optimierung enthält.

Das RO-SLAM Verfahren ist trotz der Verwendung eines Partikel Filter ein sehr ressourcenschonendes Verfahren. Bereits eine sehr geringe Anzahl an Partikeln in dem Hauptpartikel Filter führt zu einer brauchbaren Positionsschätzung der Roboterplattform. Jedoch unterscheidet sich die Güte der Positionsschätzung stark zwischen der Verwendung der realen und virtuellen UWB Module. Während die Positionsschätzung der virtuellen UWB Module zu reproduzierbaren Ergebnissen führte, verhielt es sich bei den realen UWB Modulen komplett anders. Mit jedem Durchlauf ergab sich eine komplett neue Positionsschätzung, die oftmals auch große Rotationsfehler aufwies. Ähnlich verhielt es sich bei der Positionsschätzung der UWB Module, auch hier schnitten die virtuellen UWB Module deutlich besser ab als die realen UWB Module. Der mittlere Positionsfehler der realen UWB Module betrug dabei 0,58 m und bei den virtuellen 0,39 m. Aufgrund dieser Feststellung, bleibt nur die Vermutung übrig das die Güte der Entfernungsmessung mit den realen UWB Modul nicht ausreichend ist, um eine Positionsgenauigkeit von 0,10 m zu erreichen.

Zusammenfassend kann gesagt werden, dass eine Entfernungsmessung mittels selbstgebauter UWB Module möglich ist. Die Daten dieser UWB Module können in einem bereits vorhanden RO-SLAM Softwaremodule ausgewertet werden. Jedoch, und das gilt für jeden Bereich der Robotik, muss der Algorithmus je nach Anwendungsfall entsprechend der Anforderungen angepasst werden.

---

# Literaturverzeichnis

---

- [1] Robin Murphy. *Introduction to AI robotics*. MIT press, 2000.
- [2] Sebastian Thrun, Wolfram Burgard und Dieter Fox. *Probabilistic robotics*. MIT press, 2005.
- [3] Sinan Gezici u. a. „Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks“. In: *IEEE signal processing magazine* 22.4 (2005), S. 70–84.
- [4] Hui Liu u. a. „Survey of wireless indoor positioning techniques and systems“. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 37.6 (2007), S. 1067–1080.
- [6] Reza Zekavat und R Michael Buehrer. *Handbook of position location: Theory, practice and advances*. Bd. 27. John Wiley & Sons, 2011.
- [9] Morgan Quigley u. a. „ROS: an open-source Robot Operating System“. In: *ICRA workshop on open source software*. Bd. 3. 3.2. Kobe. 2009, S. 5.
- [10] George Kantor und Sanjiv Singh. „Preliminary results in range-only localization and mapping“. In: *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*. Bd. 2. Ieee. 2002, S. 1818–1823.
- [11] Edwin Olson, John Leonard und Seth Teller. „Robust range-only beacon localization“. In: *Autonomous Underwater Vehicles, 2004 IEEE/OES*. IEEE. 2004, S. 66–75.
- [12] Jose-Luis Blanco, Javier González und Juan-Antonio Fernández-Madrigal. „A pure probabilistic approach to range-only SLAM“. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, S. 1436–1441.
- [13] Kevin Murphy und Stuart Russell. „Rao-Blackwellised particle filtering for dynamic Bayesian networks“. In: *Sequential Monte Carlo methods in practice*. Springer, 2001, S. 499–515.
- [14] Michael Montemerlo u. a. „FastSLAM: A factored solution to the simultaneous localisation and mapping problem“. In: *Aaa/iaai*. 2002, S. 593–598.
- [15] Jose-Luis Blanco, Juan-Antonio Fernández-Madrigal und Javier González. „Efficient probabilistic range-only SLAM“. In: *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE. 2008, S. 1017–1022.
- [17] Joseph Djugash und Sanjiv Singh. „A robust method of localization and mapping using only range“. In: *Experimental Robotics*. Springer. 2009, S. 341–351.

- [18] Moe Z Win und Robert A Scholtz. „Impulse radio: How it works“. In: *IEEE Communications letters* 2.2 (1998), S. 36–38.
- [19] Liuqing Yang und G. B. Giannakis. „Ultra-wideband communications: an idea whose time has come“. In: *IEEE Signal Processing Magazine* 21.6 (Nov. 2004), S. 26–54. ISSN: 1053-5888. DOI: 10.1109/MSP.2004.1359140.
- [20] Robert J Fontana. „Recent system applications of short-pulse ultra-wideband (UWB) technology“. In: *IEEE Transactions on microwave theory and techniques* 52.9 (2004), S. 2087–2104.
- [21] Roberto Aiello und Anuj Batra. *Ultra wideband systems: technologies and applications*. Newnes, 2006.
- [22] Mohammadreza Yavari und Bradford G Nickerson. „Ultra wideband wireless positioning systems“. In: *Dept. Faculty Comput. Sci., Univ. New Brunswick, Fredericton, NB, Canada, Tech. Rep. TR14-230* (2014).
- [23] V Lakkundi. „Ultra wideband communications: History, evolution and emergence“. In: *Acta Polytechnica* 46.4 (2006).
- [24] Jianli Pan. „Medical applications of ultra-wideband (UWB)“. In: *Survey Paper* (2007). URL: <http://www.cse.wustl.edu/~jain/cse574-08/ftp/uwb/>.
- [25] Amr Eltaher und Thomas Kaiser. „Positioning of robots using ultra-wideband signals“. In: *IRA workshop on Advanced control and Diagnosis*. 2004.
- [26] Terence W Barrett und VA Vienna. „Technical features, history of ultra wideband communications and radar: part I, UWB communications“. In: *Microw J* 44.1 (2001), S. 22–56.
- [27] R. G. *Transmission and reception system for generating and receiving base-band pulse duration pulse signals without distortion for short base-band communication system*. US Patent 3,728,632. 17. Apr. 1973. URL: <https://www.google.com/patents/US3728632>.
- [28] Charles Fowler, John Entzminger, JAMES Corum u. a. „Assessment of ultra-wideband (UWB) technology“. In: *IEEE Aerospace and Electronic Systems Magazine* 5.11 (1990), S. 45–49.
- [41] Andrea Censi. „An ICP variant using a point-to-line metric“. In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE. 2008, S. 19–25.
- [42] Mariano Jaimez, Javier G Monroy und Javier Gonzalez-Jimenez. „Planar odometry from a radial laser scanner. A range flow-based approach“. In: *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE. 2016, S. 4479–4485.

## Internetquellen

- [16] Jose-Luis Blanco, Juan-Antonio Fernández-Madrigal und Javier González. *Efficient probabilistic range-only SLAM*. 2008. URL: <http://ingmec.ual.es/~jlblanco/papers/iros08ro-slam.ppt>.

- [32] lady ada. *Introducing Pro Trinket. Trinket's got a big sister in town - the Pro Trinket 5V!* 20. Aug. 2014. URL: <https://learn.adafruit.com/introducing-pro-trinket>.
- [33] lady ada. *Adafruit Pro Trinket LiPoly/LiIon Backpack. Add a rechargeable battery to your Pro Trinket.* 16. Sep. 2014. URL: <https://learn.adafruit.com/adafruit-pro-trinket-lipoly-slash-liion-backpack>.
- [34] Thomas Trojer. *thotro/arduino-dw1000. A library that offers functionality to use Decawave's DW1000 chips/modules with Arduino.* 2015. URL: <https://github.com/thotro/arduino-dw1000>.
- [35] Friends of Fritzing. *fritzing. electronics made easy.* 2010. URL: <http://fritzing.org/home>.
- [36] Albert Kasdorf. *albertkasdorf/ro\_slam\_with\_uwb. Range Only SLAM with Ultra Wideband.* 2017. URL: [https://github.com/albertkasdorf/ro\\_slam\\_with\\_uwb](https://github.com/albertkasdorf/ro_slam_with_uwb).
- [43] Wikipedia. *Gleichseitiges Dreieck.* 26. Jan. 2018. URL: [https://de.wikipedia.org/wiki/Gleichseitiges\\_Dreieck](https://de.wikipedia.org/wiki/Gleichseitiges_Dreieck).
- [44] Wikipedia. *Fünfeck.* 26. Jan. 2018. URL: <https://de.wikipedia.org/wiki/F%C3%BCnfek>.

## Handbücher

- [5] *Real time location systems - An Introduction.* DecaWave Limited, 2014. URL: [https://www.decawave.com/sites/default/files/aps003\\_dw1000\\_rtls\\_introduction.pdf](https://www.decawave.com/sites/default/files/aps003_dw1000_rtls_introduction.pdf) (besucht am 13.12.2017).
- [7] *The implementation of two-way ranging with the DW1000.* Version 2.2. DecaWave Limited, 2015. URL: <https://www.decawave.com/application-notes>.
- [8] *DWM1000 User Manual.* Version 2.10. DecaWave Limited, 2016. URL: <https://www.decawave.com/application-notes>.
- [29] *UWB Regulations. A Summary of Worldwide Telecommunications Regulations governing the use of Ultra Wideband.* Version 1.0. DecaWave Limited, 2015. URL: [https://www.decawave.com/sites/default/files/apr001\\_uwb\\_worldwide\\_regulations\\_summary.pdf](https://www.decawave.com/sites/default/files/apr001_uwb_worldwide_regulations_summary.pdf) (besucht am 13.12.2017).
- [30] *DW1000 Datasheet.* Version 2.12. DecaWave Limited, 2016. URL: <https://www.decawave.com/application-notes>.
- [31] *DWM1000 Datasheet.* Version 1.6. DecaWave Limited, 2016. URL: <https://www.decawave.com/application-notes>.
- [37] *Antenna delay calibration of DW1000-based products and systems.* Version 1.01. DecaWave Limited, 2014. URL: [https://www.decawave.com/sites/default/files/aps014\\_antennadelaycalibrationofdw1000-basedproductsandsystems\\_v1.01.pdf](https://www.decawave.com/sites/default/files/aps014_antennadelaycalibrationofdw1000-basedproductsandsystems_v1.01.pdf) (besucht am 13.12.2017).
- [38] *Robotino Handbuch.* Festo Didactic, 2007. URL: [http://www.festo-didactic.com/ov3/media/customers/1100/544305\\_robotino\\_deen.pdf](http://www.festo-didactic.com/ov3/media/customers/1100/544305_robotino_deen.pdf).

- [39] *TiM55x / TiM56x / TiM57x Betriebsanleitung*. SICK, 2016. URL: <https://www.sick.com/de/de/mess-und-detektionsloesungen/2d-lidar-sensoren/tim5xx/tim571-2050101/p/p412444>.
- [40] *Product Brief NUC5i7RYH*. Intel, 2015. URL: <https://www.intel.de/content/www/de/de/nuc/nuc-kit-nuc5i7ryh-brief.html>.

# **Anhang**



# A.

## Abbildungen

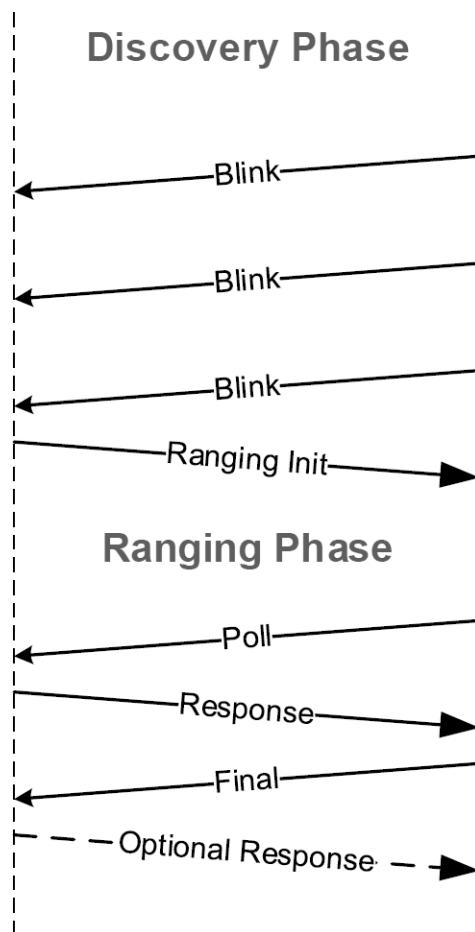


Abbildung A.1.: Der Ablauf der Discovery- und Range-Phase zwischen einem Tag und mehreren Ankern. [8]

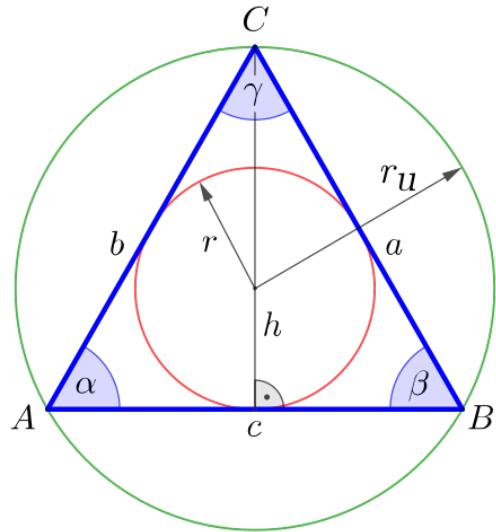


Abbildung A.2.: Ein gleichseitiges Dreieck. [43]

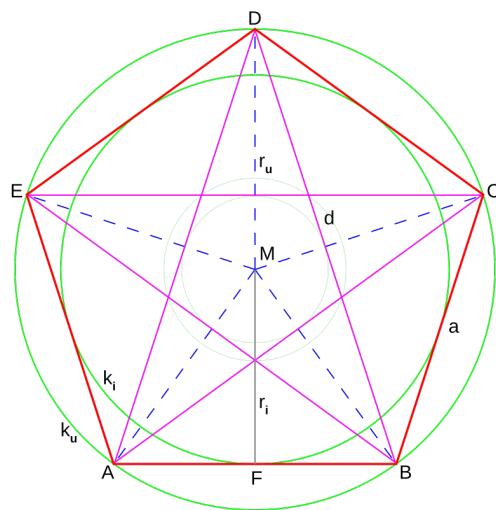


Abbildung A.3.: Ein regelmäßiges Fünfeck und ein Pentagramm. [44]

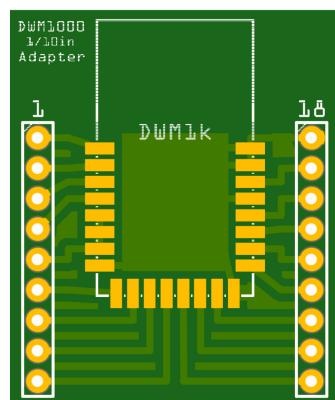


Abbildung A.4.: PCB-Adapterboard für den DWM1000.

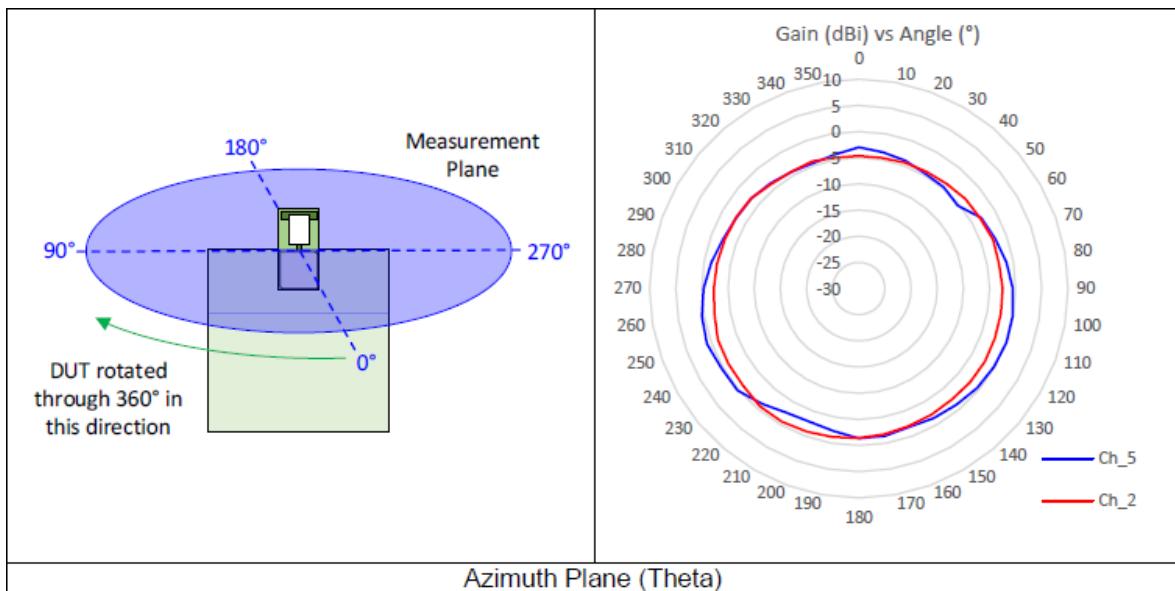


Abbildung A.5.: Das Antennenabstrahlmuster in horizontaler Richtung. [31]

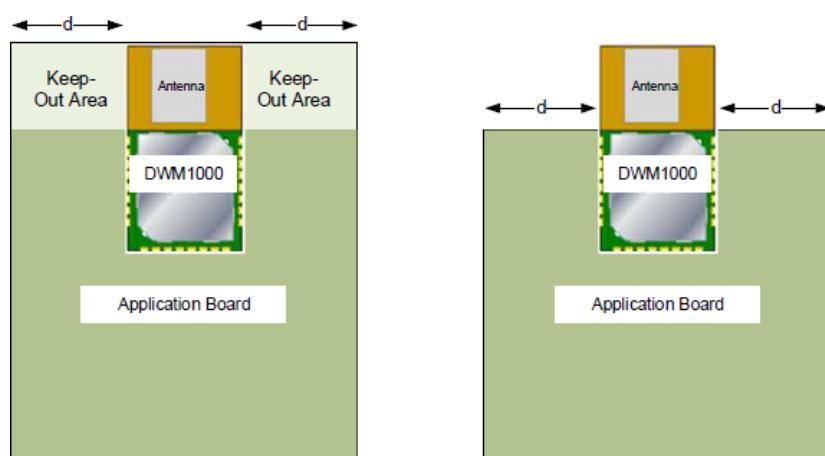


Abbildung A.6.: Antennenfreiräume [31]

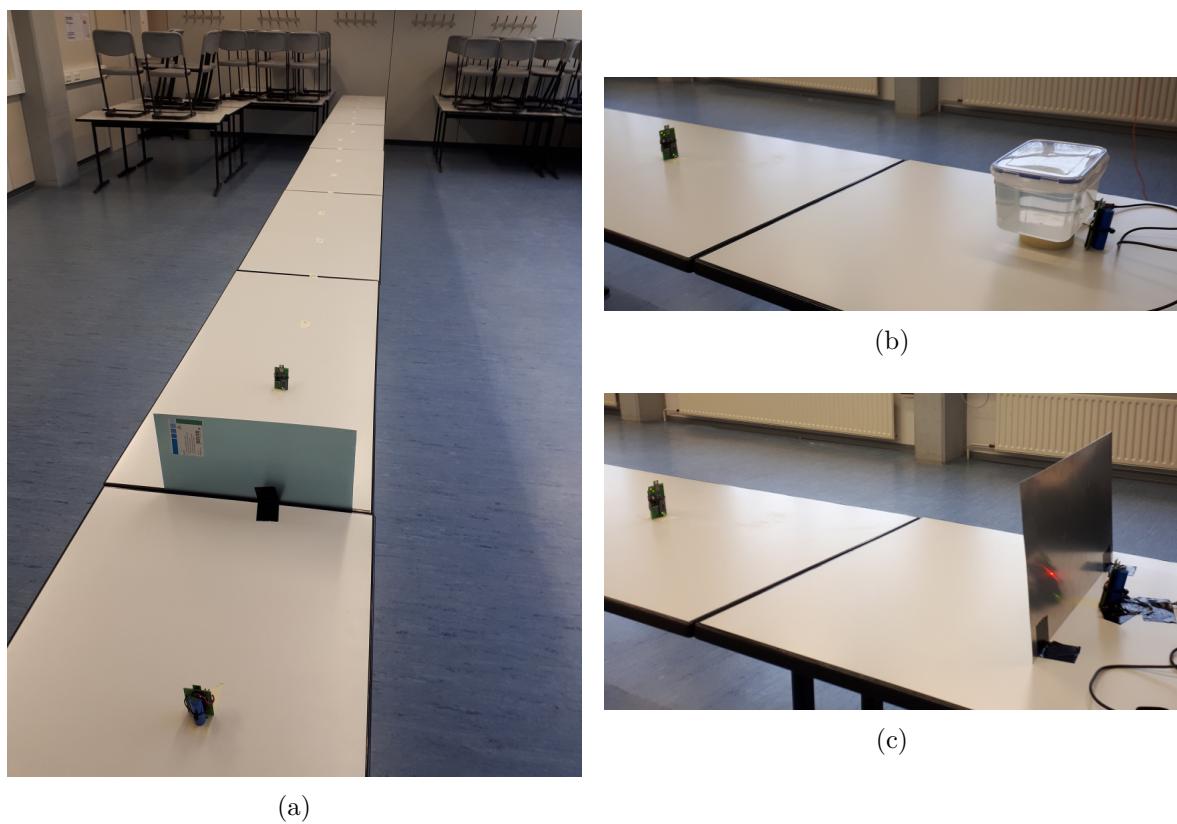


Abbildung A.7.: Versuchsaufbau für die NLoS-Entfernungsmessung.

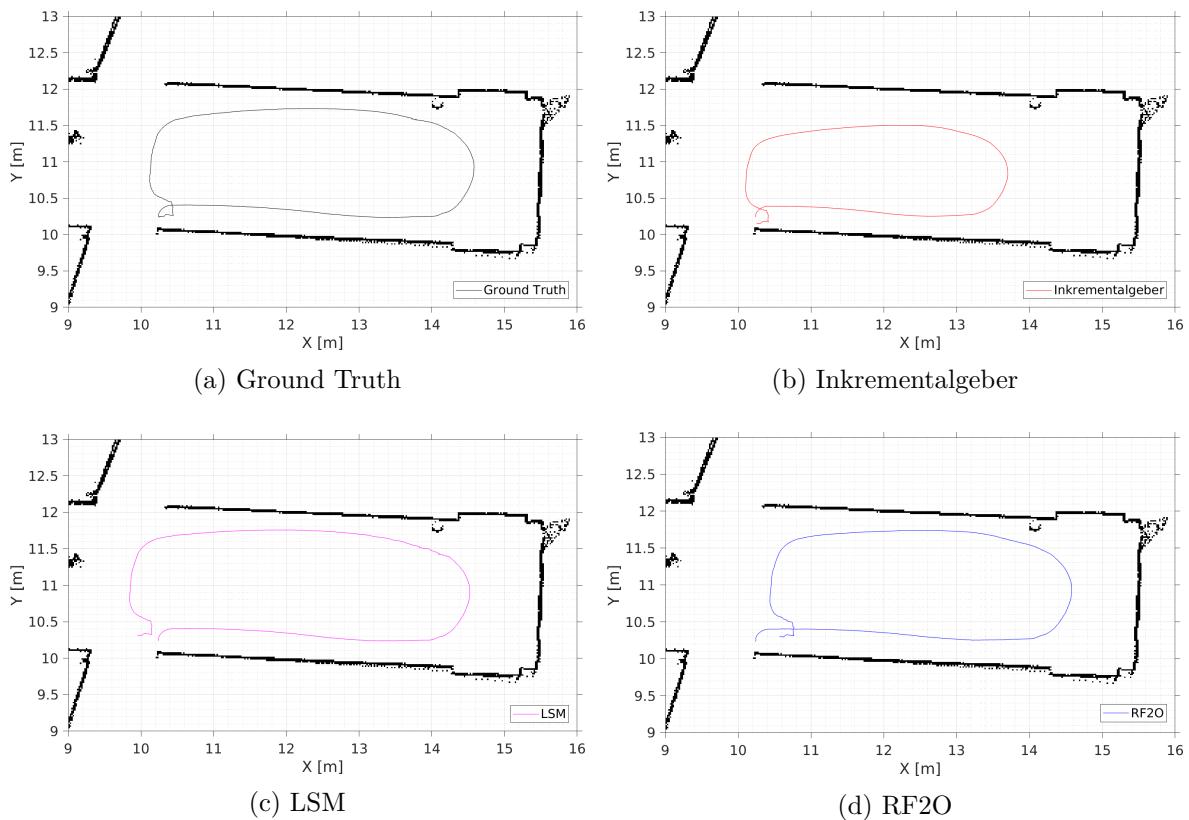


Abbildung A.8.: Rechteckige Trajektorien der verschiedenen Odometriequellen.

# B.

## Tabellen

DWM1000 (Pin)	Pro Trinket (Pin)	Bedeutung
SPICLK (20)	SCK (13)	SPI
SPIMISO (19)	MISO (12)	SPI
SPIMOSI (18)	MOSI (11)	SPI
SPICSn (17)	SS (10)	SPI
IRQ (22)	INT1 (3)	Interrupt
RSTn (3)	PB1 (9)	Hardware Reset

Tabelle B.1.: Pinbelegung zwischen dem DWM1000 und Pro Trinket.

LiIon Backpack	Pro Trinket	Bedeutung
BAT	BAT+	Batteriespannung
5V	BUS	Ladespannung
G	GND	Masse
SW1	–	Schalter
SW2	–	Schalter

Tabelle B.2.: Pinbelegung zwischen dem LiIon Backpack und dem Pro Trinket.

CP2104 Friend	Pro Trinket	Bedeutung
GND	GND	Masse
CTS	GND	Masse
5V	5V	Versorgungsspannung
TXD	RXD	Datenaustausch
RXD	TXD	Datenaustausch
RTS	RTS	Reset

Tabelle B.3.: Pinbelegung zwischen dem CP2104 Friend und dem Pro Trinket.

Tag	Anker	Anbieter	Artikelnummer	Komponente	Anzahl	Stückpreis
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Digi-Key	1479-1002-1-ND	Decawave Limited DW1000	1	21,23 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Digi-Key	1528-1040-ND	Adafruit Pro Trinket 3 V	1	8,25 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Conrad	1417697-62	Kohleschicht-Widerstand 10kΩ	1	0,06 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Conrad	1417644-62	Kohleschicht-Widerstand 180Ω	4	0,06 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Conrad	180134-62	LED Grün	2	0,25 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Conrad	184756-62	LED Rot	2	0,34 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Conrad	251007-62	Emmerich Lilon Akku ICR-18650 NH-SP	1	9,99 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EXP-Tech	EXP-R15-577	Adafruit Pro Trinket LiPoly/Lilon Backpack	1	5,55 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EXP-Tech	EXP-R15-1157	Adafruit CP2104 Friend	1	6,45 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EXP-Tech	EXP-R15-419	Breadboard-friendly SPDT Slide Switch	1	1,20 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	EXP-Tech	EXP-R15-1074	JST 2-Pin Cable	1	0,80 €
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	AISLER	-	Printed Circuit Board	1	7,74 €
Gesamtpreis für einen Tag:						45,15 €
Gesamtpreis für einen Anker:						56,24 €

Tabelle B.4.: Materialkosten pro Tag bzw. Anker.

Entfernung [m]	$\bar{x}$ [m]	$\sigma$ [m]	$SE_{\bar{x}}$ [m]	Min [m]	Max [m]	Bias [m]
1,0	1,222	0,018	0,0008	1,130	1,270	0,222
1,5	1,760	0,017	0,0007	1,680	1,820	0,260
2,0	2,262	0,019	0,0008	2,220	2,350	0,262
2,5	2,729	0,018	0,0008	2,670	2,790	0,229
3,0	3,240	0,021	0,0009	3,190	3,300	0,240
3,5	3,758	0,021	0,0009	3,710	3,810	0,258
4,0	4,210	0,019	0,0008	4,150	4,260	0,210
4,5	4,702	0,018	0,0008	4,640	4,760	0,202
5,0	5,182	0,019	0,0008	5,120	5,260	0,182
5,5	5,690	0,020	0,0009	5,630	5,750	0,190
6,0	6,205	0,021	0,0009	6,120	6,270	0,205
6,5	6,714	0,022	0,0010	6,660	6,810	0,214
7,0	7,227	0,025	0,0011	7,160	7,310	0,227
7,5	7,735	0,026	0,0011	7,670	7,810	0,235
8,0	8,209	0,023	0,0010	8,140	8,280	0,209
8,5	8,707	0,023	0,0010	8,630	8,770	0,207
9,0	9,204	0,023	0,0010	9,120	9,290	0,204

Tabelle B.5.: Stochastische Merkmale der LoS–Entfernungsmeßung.

Entfernung [m]	$\bar{x}$ [m]	$\sigma$ [m]	$SE_{\bar{x}}$ [m]	Min [m]	Max [m]	Bias [m]
1,0	1,617	0,164	0,0073	1,170	2,160	0,617
1,5	2,026	0,193	0,0086	1,520	2,340	0,526
2,0	2,471	0,192	0,0086	2,120	2,900	0,471
2,5	2,681	0,042	0,0019	2,590	3,030	0,181
3,0	3,139	0,039	0,0017	3,060	3,500	0,139
3,5	3,796	0,154	0,0069	3,580	4,420	0,296
4,0	4,384	0,215	0,0096	4,060	5,000	0,384
4,5	4,928	0,212	0,0095	4,550	5,940	0,428
5,0	5,786	0,111	0,0050	5,310	5,940	0,786
5,5	6,530	0,263	0,0118	5,800	7,140	1,030
6,0	6,879	0,054	0,0024	6,740	7,300	0,879
6,5	7,364	0,060	0,0027	7,240	7,850	0,864
7,0	7,819	0,076	0,0034	7,690	8,690	0,819
7,5	8,466	0,193	0,0086	8,210	9,750	0,966
8,0	8,907	0,125	0,0056	8,750	9,540	0,907
8,5	9,626	0,340	0,0152	9,190	11,010	1,126
9,0	10,680	0,315	0,0141	9,980	11,680	1,680

Tabelle B.6.: Stochastische Merkmale der NLoS–Entfernungsmeßung mit einem wassergefüllten Kunststoffbehälter.

Entfernung [m]	$\bar{x}$ [m]	$\sigma$ [m]	$SE_{\bar{x}}$ [m]	Min [m]	Max [m]	Bias [m]
1,0	1,285	0,021	0,0009	1,230	1,360	0,285
1,5	1,770	0,019	0,0008	1,710	1,840	0,270
2,0	2,305	0,020	0,0009	2,240	2,360	0,305
2,5	2,803	0,020	0,0009	2,740	2,870	0,303
3,0	3,303	0,022	0,0010	3,250	3,370	0,303
3,5	3,816	0,022	0,0010	3,750	3,880	0,316
4,0	4,269	0,024	0,0011	4,170	4,340	0,269
4,5	4,728	0,021	0,0010	4,670	4,860	0,228
5,0	5,250	0,023	0,0010	5,180	5,320	0,250
5,5	5,707	0,023	0,0010	5,630	5,780	0,207
6,0	6,276	0,032	0,0014	6,150	6,360	0,276
6,5	6,801	0,029	0,0013	6,730	6,890	0,301
7,0	7,253	0,040	0,0018	7,150	7,560	0,253
7,5	7,788	0,055	0,0025	7,650	8,140	0,288
8,0	8,290	0,030	0,0013	8,200	8,400	0,290
8,5	9,066	0,379	0,0169	8,630	10,280	0,566
9,0	9,490	0,237	0,0106	9,110	10,250	0,490

Tabelle B.7.: Stochastische Merkmale der NLoS–Entfernungsmessung mit einem Aluminiumblech in einer Entfernung von 50 cm.

Entfernung [m]	$\bar{x}$ [m]	$\sigma$ [m]	$SE_{\bar{x}}$ [m]	Min [m]	Max [m]	Bias [m]
1,0	1,722	0,234	0,0105	1,320	2,350	0,722
1,5	1,744	0,056	0,0025	1,660	2,410	0,244
2,0	2,260	0,028	0,0012	2,190	2,360	0,260
2,5	2,800	0,025	0,0011	2,730	2,870	0,300
3,0	3,284	0,025	0,0011	3,210	3,390	0,284
3,5	3,803	0,034	0,0015	3,730	4,290	0,303
4,0	4,285	0,029	0,0013	4,210	4,380	0,285
4,5	4,754	0,033	0,0015	4,660	5,030	0,254
5,0	5,320	0,166	0,0074	5,140	6,370	0,320
5,5	6,113	0,381	0,0170	5,690	7,790	0,613
6,0	6,538	0,339	0,0152	6,160	7,700	0,538
6,5	7,311	0,382	0,0171	6,650	8,760	0,811
7,0	8,531	0,342	0,0153	7,500	9,190	1,531
7,5	8,783	0,285	0,0128	8,020	9,750	1,283
8,0	9,543	0,077	0,0034	9,190	10,300	1,543
8,5	10,089	0,230	0,0103	9,480	10,740	1,589
9,0	10,622	0,353	0,0158	10,160	11,500	1,622

Tabelle B.8.: Stochastische Merkmale der NLoS–Entfernungsmessung mit einem Aluminiumblech in einer Entfernung von 5 cm.